CASIO®

はじめに

本書はこれからBASICプログラムを始めようとする方にも、すでにBASICを知っていて、すぐにPB-110をフル活用しようとする方にも、わかりやすく説明しています。特に、これから始めようとする方にとって、よりわかりやすく、楽しみながらBASICの勉強にも役立つよう楽しいゲームを中心に書かれています。

まずは難しいことは抜きにして、ゲームを楽しみながらBASICプログラムとはどういうものか、どのように組み立てられているかを覚えてください。BASICで使われる各命令は、少しづつ、順に覚えていけば簡単なものです。

最初からあせらずに、ゆっくりとお楽しみください。

本書は次のような7章より構成されております。

第1章　覚えておこう本体構成と使い方
第2章　PB-110を動かそう
第3章　BASICプログラムとは
第4章　実践プログラム
第5章　コマンド・リファレンス
第6章　PB-110活用編
第7章　ライブラリー

BASICプログラムを初めて習う方は第1章から順にお読みになり、プログラムの基本をしっかりと覚えてください。特に第3章、第4章ではプログラムの流れにそって説明しております。

すでにBASICを知っている方は第1章、第2章で基本的な操作を覚えた後は、第5章のコマンドリファレンスを読みながら、PB-110の特長をつかんでください。

なお、すぐにプログラムを入力して使いたい方は第4章、第7章のプログラム例をお使いになったり、第6章を参考にPB-100シリーズのプログラムを使ってみてください。

では、この本を参考にしてPB-110を有効にお使いください。

ご使用の前に

この計算機は、カシオの高度な電子技術と品質管理のもとで、厳重な検査工程を経て、皆様のお手もとに届けられています。
本機を末ながくご愛用いただくために、次の点にご留意のうえ、お取り扱いください。

■ご使用上の注意

- 計算機は精密な電子部品で構成されていますので、絶対に分解しないでください。また、投げたり落したり等のショックや、急激な温度変化を与えないでください。特に、高温の所、湿気やホコリの多い所に放置したり保管することはしないでください。なお、温度が低いときは表示の応答速度が遅くなったり、点灯しなくなることがありますが、通常の温度になると正常にもどります。
- アダプター差し込み口には、FA-3およびFP-12またはFP-12S以外は接続しないでください。
- 計算機の演算中は"−"を表示し、この間のキー操作は一部キーを除いて無効ですから、常に表示を確認しながら、確実にキーを押してください。
- ブザーを鳴らしたときに表示が薄くなることがありますが、故障ではありません。なお、あまり表示が薄いときは、早めに電池を交換してください。
- 電池は、使わない場合でも2年に1度は交換してください。
  特に消耗ずみの電池を放置しておきますと、液もれをおこし、故障等の原因になりますので、計算機内には絶対に残しておかないでください。
- アダプター差し込み口のキャップは、本体のみで使用する場合には必ずつけて、むやみに接点には触れないでください。
- 本体が強度の静電気をおびますと、メモリー内容が変化したり、キー操作ができなくなることがあります。このような場合には、一旦電池をはずし、もう一度入れなおしてください。
- オプションとの接続は本体の電源スイッチをOFFにしてから行なってください。
- 計算機のお手入れは、シンナー・ベンジン等の揮発性液体をさけ、「乾いた柔らかい布」あるいは、「中性洗剤液に浸し固くしぼった布」でおふきください。
- プログラム実行中または演算中には、電源スイッチを切らないでください。
- 本機は高精密機器のため、プログラム実行中に**強い振動や衝撃**を与えますと、プログラム実行が停止したり、メモリーの内容が変化する場合がありますのでご注意ください。

■保証・アフターサービス

- 保証は、別紙の保証書の内容によりますので、よくお読みのうえ、記入事項を確認して、大切に保管してください。
- 万一故障したときは　**❶お買い上げ店**　**❷カシオ計算機サービスセンター**のうち、ご都合のよい所へ、必ず保証書をそえて、ご持参またはご郵送ください。この場合、故障内容を具体的にお知らせください。
- 修理依頼される前には、この説明書をもう一度お読みになると共に、電源の状態および、プログラムミス、操作ミスがないかをよくお調べください。
- ご不明の点やご質問、お問い合わせ等は、171ページのカシオ計算機へ直接ご連絡ください。

目　次

目　　次

第4章　実践プログラム　47

第5章　コマンド・リファレンス　85

目　　次

目　　次

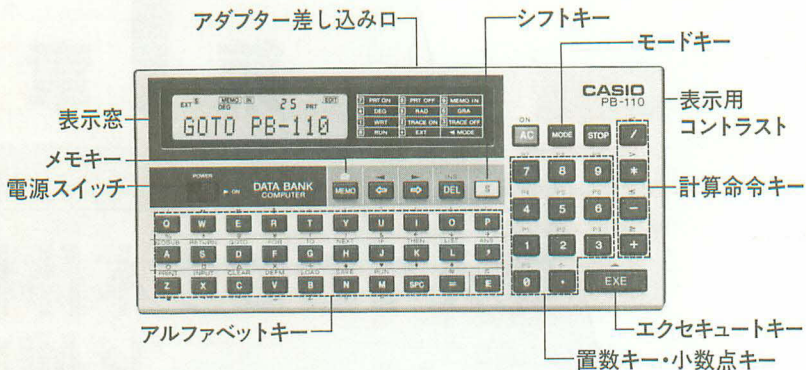
目　　次

巻末資料

第1章

覚えておこう本体構成と使い方

今迄にコンピュータに触れたことのない方も、もうすでにコンピュータに慣れた方も、まずこの章はよくお読みください。本機の本体構成や使い方を覚えていただくのが、早く使いこなすコツです。

1−1 各部の名称とそのはたらき

普通の電卓に比べて、たくさんのキーや差し込み口があります。これだけキーがたくさんあると、どのキーを何に使うのかと迷われるかもしれません。この迷いを取り除くために、これから各々のキーや差し込み口などを説明します。

●**電源スイッチ**

スライド式のスイッチで、右にスライドすると電源が入り、左にスライドさせると電源が切れます。

●**シフトキー(赤色のSキー)**

パネル上に赤色で書かれているワンキーコマンドや記号を表示させるときに押します。一度押しますとシフトインモードとなり"S"が点灯します。続けて押すとシフトインモードが解除され、"S"が消えます。
(本書ではアルファベットキーのSと区別するために、以後SHIFTと書きます)

●置数キー・小数点キー、計算命令キー、エクセキュートキー

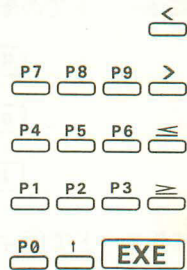
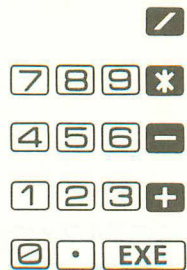
このキーの配列をよく見てください。普通の電卓と同じ配列をしていますね。この部分はちょうど四則計算(加減乗除)をするときに使いますが、少し異なる点があります。それは⊠(カケル)と⊞(ワル)のキーがちがっていることと、⊟(イコール)キーがなく、EXE(エクセキュート)というキーがあります。これはコンピュータの言葉では×は＊(アスタリスク)を、÷は／(スラッシュ)を使い、⊟キのかわりにEXEキーで答えを求めます。

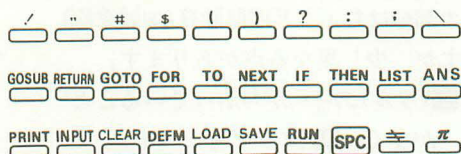
/
7 8 9 ＊
4 5 6 −
1 2 3 ＋
0 ・ EXE

例えば、普通の電卓で 12 × 4 ÷ 3 ＋ 7 − 5 ＝ と操作するところを、
本機では、12 ＊ 4 / 3 ＋ 7 − 5 EXE と操作します。
これで、本機を普通の電卓がわりに使うこともできます。
なお、SHIFTキーに続けて押しますと、0〜9キーはP0〜P9のプログラムエリア指定の役目をし、・キーはべき乗計算(x^y → x ↑ y)の、＋−＊／キーは大小比較記号(≧、≦、>、<)を表示します。

<
P7 P8 P9 >
P4 P5 P6 ≦
P1 P2 P3 ≧
P0 ↑ EXE

●アルファベットキー、スペースキー

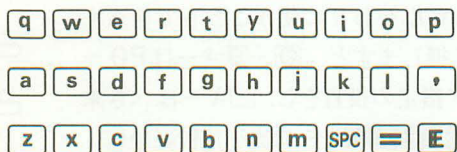
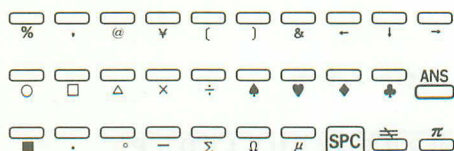
Q W E R T Y U I O P
A S D F G H J K L ,
Z X C V B N M SPC ＝ E

このキーが本機の特長で、タイプライターのようにアルファベット26文字と、スペースキー(SPC)が並んでいます。このキーを使って命令を与えたり、プログラムを書き込んだりします。また、A〜Zまでの26文字のキーはそれぞれがメモリー(記憶するところ)の役をします。
なお、A〜Zのキーには別の役割があり、SHIFTキーに続けて押すと記号やBASICのコマンドを表示します。

例） [SHIFT][A]→GOSUB、 [SHIFT][U]→？

このアルファベットキーはほかにも使い方があります。それは拡張モード([MODE]キーに続いて[・]キーを押す。EXT点灯)での使い方で、直接押すとアルファベットの小文字を、[SHIFT]キーに続けて押すと特殊記号を表示します。

拡張モードでの働き

[q][w][e][r][t][y][u][i][o][p]

[a][s][d][f][g][h][j][k][l][,]

[z][x][c][v][b][n][m][SPC][=][E]

拡張モードで[SHIFT]キーに続けて押したときの働き

拡張モードを解除して、アルファベット大文字に戻すには、もう一度[MODE][・]と押します。
このように、アルファベットキーはいくつもの顔を持っていますので、よく覚えておいてください。

● **イコールキー([=])**

このキーは計算の答を求めるためのキーではなく、代入文(55ページ参照)やIF文(51ページ参照)での判断のために使います。
なお、[SHIFT]キーに続いて押しますと、≠(等しくない)の記号が表示されます。

● 指数部置数キー・パイキー(π/E)

このキーは直接押すと指数部置数キーとなり、指数部(10の何乗)を置数する前に押します。例えば1.23×10^{4}の場合は[1][・][2][3][E][4]と押します。 指数部が負の場合は、このキーに続いて[−]キーを押します。7.41×10^{-9}は [7][・][4][1][E][−][9]と押します。

[SHIFT]キーに続いて押しますとπ(パイ……円周率)を表示します。

● アンサーキー(ANS/・)

このキーは[SHIFT]キーに続けて押しますと直前に行なわれた計算の答を覚えているキーで、マニュアル計算とプログラム計算で行なわれた計算の答を表示します。

● モードキー([MODE])

このキーは計算機の状態や角度単位を指定するするときに、[・][0]〜[9]のキーと組み合わせて使います。

[MODE][・]……"EXT"を表示し、拡張モードとなり、英小文字・特殊記号が使えます。再び押すと拡張モードを解除します。

[MODE][0]……"RUN"を表示し、マニュアル計算およびプログラム計算が行なえます。

[MODE][1]……"WRT"を表示し、プログラムの書き込みおよびチェック、編集が行なえます。

[MODE][2]……"TR"を表示し、実行トレースが行なえます。(詳しくは41ページ)

[MODE][3]……"TR"を表示している場合は"TR"が消え、実行トレース機能が解除されます。

[MODE][4]……"DEG"を表示し、角度の単位を〈度〉に指定します。

[MODE][5]……"RAD"を表示し、角度の単位を〈ラジアン〉に指定します。

[MODE][6]……"GRA"を表示し、角度の単位を〈グレード〉に指定します。

[MODE][7]……"PRT"を表示し、プリンタ接続時はプリント出力をすることができます。

[MODE][8]……"PRT"が表示している場合は"PRT"が消え、プリント出力が解除されます。

[MODE][9]…… "[MEMO][IN]"を表示し、メモインモードとなります。(データバンク活用法参照)このモードを解除するには、[MODE][0]と操作します。

●メモキー(MEMO)
データバンクを使うときに押します。RUNモード(MODE 0と押す)やメモインモード(MODE 9と押す)で直接押して順番に呼び戻すか、文字を指定した後に押して呼び戻すときに使います。

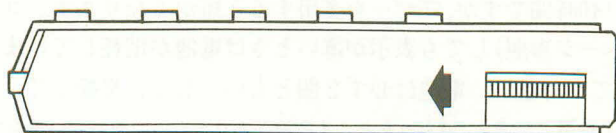
●カーソルキー(◀▶)
このキーは表示されている文字を訂正するときに便利なキーで、カーソル(表示窓で点滅している"_"のことです) を左右に移動させます。1回押すと一文字分移動し、押し続けますと文字の書いてある範囲内を続けて移動します。

●オールクリアーキー(AC)
このキーは全てのキーの中で一番強いキーで、どんな表示でも消してしまいます。また、エラーになって停止したときやオートパワーオフ（9ページ参照）で表示が消えているときにも押します。プログラム実行中は、プログラムを中断させます。

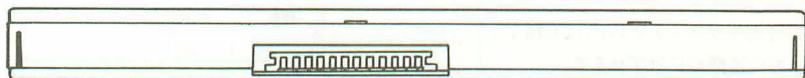
●デリート・インサートキー(INS/DEL)
表示されている文字を訂正するときに便利なキーで、カーソルが点滅している文字を削除(デリート)します。削除した後はカーソルより右側の文字を左につめます。
また、SHIFTキーに続けて押しますと、カーソルの点滅している文字以降を右にずらし、空白をあけます。

●ストップキー(STOP)
このキーはプログラム実行中に使うキーで、プログラムの実行を一時的に停止させます。プログラムを続けて実行させたいときはEXEキーを押すことにより再開します。
プログラム停止中に押しますと、プログラムエリア番号と行番号を表示します。
表示がスクロール中(13ページ参照)では、スクロールを停止します。スクロールを続けるときはEXEキーを押します。

●**表示用コントラスト**

電池の消耗や表示窓を見る角度により、濃く見えたり薄く見えたりします。このようなときには、本体右側面にあるボリウムで、見やすい濃さに調整してください。

ボリウムは矢印方向に回すと濃くなり、逆に回すと薄くなります。
なお、最も濃い位置にしてもまだ表示が薄い場合は、電池がかなり消耗していることが考えられますので、なるべく早く電池を交換してください。

●**オプション差し込み口**

この差し込み口は別売のオプションを接続するコネクターで、プリンタを使うときは〈FP-12〉または〈FP-12S〉を、テープレコーダーで記録するときには〈FA-3〉をつなぎます。

この差し込み口には〈FP-12〉または〈FP-12S〉、〈FA-3〉以外は接続しないでください。
また、オプションを接続しないときは、必ず付属のコネクターカバーをつけて使用してください。

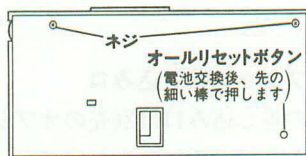
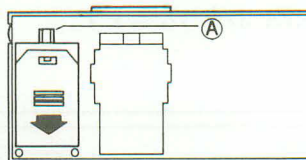
1−2 電源について

本機は2個のリチウム電池(CR-2032)を電源としています。電池の寿命は本体のみ使用で約140時間ですが、ブザーを多用すると短かくなります。コントラストを調整(7ページ参照)しても表示が薄いときは電池が消耗していますので、早めに交換してください。電池は必ず2個ともいっしょに交換してください。

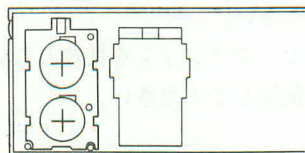
※電池は2年以上使用した場合、液もれをおこす危険がありますので使わない場合でも2年に1度は必ず電池交換してください。

※本体に組み込まれている電池はモニター用の電池ですので所定の時間に満たないうちに消耗することがあります。なるべく早目に交換してください。

■電池交換の仕方

(1)電源スイッチを切ってから、裏面の2個のネジをはずし、裏ブタをはずします。

(2)図のⒶを押しながら矢印方向にスライドさせて、電池押さえ板をはずします。

(3)古い電池を2個とも取り出します。
(電池ボックスを下に向けて軽くたたけば簡単にはずれます。)

(4)新しい電池の表面を乾いた布でよくふいてから⊕側を上にして入れます。

(5)電池押さえ板で電池を押さえながらスライドさせてとじます。

(6)裏ブタをネジ止めし、電源スイッチをONにしてからリセットボタンを先の細い棒で押してください。

※ブザーを多用しますと電池寿命は短かくなります。
※電池は必ず2個とも交換してください。
※消耗ずみの電池は絶対に火中に投入しないでください。破裂することがあり、非常に危険です。
※電池の⊕⊖は絶対にまちがえないようにしてください。

電池は、幼児の手のとどかないところに保管してください。
万一、飲み込んだ場合にはただちに医師と相談してください。

■オートパワーオフ(自動電源OFF)

スイッチの切り忘れによるムダな電力消費を防ぐ自動節電機能で、操作完了後(プログラム計算中を除く)約6分で自動的に電源オフになります。
この場合は、電源スイッチを入れ直すか、ACキーを押せば、再び電源オンになります。

※電源オフになってもメモリー内容およびプログラム内容は消えませんが、角度指定や各モード指定("WRT"、"TR"、"PRT"等)はすべて初期状態(電源スイッチをONにしたとき)となります。

1−3 増設RAMパックについて

本機のRAMエリア(メモリー)は標準で544ステップ、26メモリーですが、別売のRAMパック〈OR-1Ⓔ〉を増設することにより、最大1,568ステップに増やすことができます。
この増設されたRAMエリアは、標準のエリアと同様に使え、ステップ数の増量や、メモリーの増設(11ページ参照)を行なうことができます。

■RAMパックの取り付け方

〈準備〉

パックを取り扱うときは静電気により内部回路が破壊されることがありますのでパックを取り扱う前にドアのノブなど、金属物に手を触れて、体にたまった静電気を放電させておいてください。

〈手順〉

(1)電源を切ります。
(電源スイッチ→OFF)

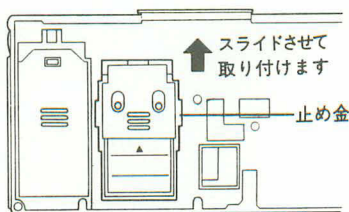
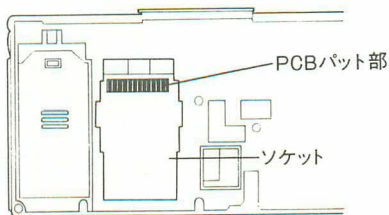
(2)本体裏側のネジを2本ともはずし、裏ブタをはずします。

(3)RAMパックについてる止め金を下にさげて、パックを本体のソケットに入れ、止め金を押さえながらスライドさせます。

※RAMパックのコネクター部、PCBパット部には絶対手を触れないでください。

(4)裏ブタをネジ止めします。

(5)電源スイッチをONにし、本体裏面のオールリセットボタンを先の細い棒などで押します。

★RAMパックを本体に増設したり、取りはずしたりした後で、オールリセットボタンを押さない場合は、メモリー内容が変化したり、無意味な表示が現われることがあります。
★パックのコネクターおよびPCBパット部にゴミ、ホコリ、指紋などがつきますと接続不良の原因となりますので、絶対に触れないようにしてください。
★取りはずしたパックは必ずケースに入れ、ゴミ、ホコリのつかないように保管してください。

■メモリーの増設

メモリー(変数)は通常A～Zまでの26メモリーで、そのときのステップ数は544ステップとなります。

このメモリーは、標準で最大94個、RAMパック増設時で最大222個まで増設することができます。このメモリーの増設は、1メモリーにつき8ステップの換算でプログラムステップをメモリーに交換します。

メモリー数		26個	27個	28個	～	46個	～	94個	～	200個	～	222個
プログラムステップ数	標準時	544ステップ	536ステップ	528ステップ	～	384ステップ	～	0ステップ	～	—		—
	増設時	1568ステップ	1560ステップ	1552ステップ	～	1408ステップ	～	1024ステップ	～	176ステップ	～	0ステップ

メモリーの増設は1個単位で、DEFMコマンドを使い行ないます。

〈例〉

Z(1)～Z(30)までの30個を増設して56個とします。

〈操作〉

モードはRUNモード(MODE 0と押す)またはWRTモード(MODE 1と押す)で行ないます。

DEFM 30 EXE　　　***VAR:56

※DEFMはDEFMと押してもSHIFT DEFMと押しても同じです。

現在メモリー数がいくつに設定されているかを確認するときもDEFMコマンドを使います。

〈例〉

メモリー数が合計で56個に設定されています。

DEFM EXE　　　***VAR:56

〈例〉

メモリー数を初期の26個に戻すには増設0を指定します。

DEFM 0 EXE　　　***VAR:26

★すでに相当数のプログラムステップが使用されている場合には、書き込みずみのプログラムを保護するために、ステップ数不足となる指定はエラー(ERR1…ステップ数不足)となります。

★専用文字変数($)は、特別メモリーのため、指定時には数えません。

★DEFMコマンドはプログラム中に書き込んで使うことができますので、個別のプログラムに合わせて書き込むと便利に使えます。

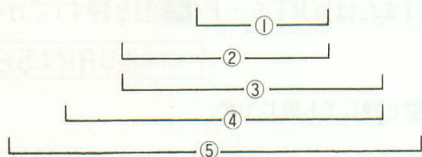
1−4 計算の前に

■計算の優先順位

計算には「優先順位」という規則があり、たし算・ひき算よりかけ算・わり算の方を先に計算することになっています。本機はこの優先順位を計算機自身が自動的に判別するようにできています。この機能はとても便利で、数式をそのまま打ち込んでいけば正しい答が求められます。
計算の優先順位は次のように定められています。

①関数（SIN, COS等）
②べき乗（↑）
③×(＊)、÷(／)
④+、−

計算はこの優先順位に従って行なわれますが、優先順位が同じときは頭から(左から)、カッコが使用されたときはカッコの内が最優先されます。

〈例〉　2＋3＊SIN（17＋13）↑2＝2.75

■入出力桁数と演算桁数

本機の数値を入力できる範囲(入力桁数)は仮数部12桁、指数部2桁です。内部演算も同じで、仮数部12桁、指数部2桁で行なわれます。
数値を表示できる範囲(出力桁数)は通常、仮数部10桁で、マニュアル計算による答の表示とプログラム中での答の表示は異なります。マニュアル計算では指数部や負符号がつく場合、仮数部、指数部、負符号を含めて12桁まで表示します。プログラム中では仮数部10桁、指数部2桁を全て表示しますが、合計が12桁をこえる場合、最初に頭から12桁分を表示し、以後順に表示が左に送られるようにして(スクロール)表示されます。

〈例〉

マニュアル計算

操作	表示
1 [・] 2345678912 [EXE]	1.234567891
12345678912 [＊] 100 [EXE]	1.2345678 E12
12345678912 [＊] −100 [EXE]	−1.234567 E12

プログラム計算

PRINT 12345678912 [＊] −100 の場合

表示の外に消えます。　　まだ表示されていません。

第2章

PB-110を動かそう

ここでは、本機に慣れ親んでいただくために、とにかくさわってください。多少まちがった操作をしても壊れるものではありません。習うより慣れろのことわざ通りに、まずは簡単な操作から。

2−1 とにかくさわってみよう

ここではまずさわって動かしてみて、どのように動くかを見てみましょう。
まず最初にすることは、本機を手に取って、電源スイッチを右にスライドさせて、電源を入れます。

すると表示は次のようになるはずです。

RUN
DEG
READY P0

初めにこの表示を消してみます。[AC]キーを押してください。"READY P0"は消えましたね。このとき左端に点滅している"_"があります。これを**「カーソル」**と呼び、ここから文字を書き始めます。

このカーソルが点滅している状態を**「入力待ち状態」**とも言い、計算や命令を待っているのです。カーソルは通常"_"の点滅ですが、文字を続けて書いていくうちに、"▮"の点滅となることがあります。本機では1行に書ける文字数が62文字までですので、56文字以上書きますと注意信号として現われます。なお、表示窓の上に"RUN"と"DEG"が点灯していると思いますが、これは状態表示といい、今どのような状態になっているかを示します。"RUN"はRUN(ラン)モードを示し、マニュアル計算やプログラムの実行が行なえます。"DEG"は角度単位で、度になっていることを示します。角度単位はほかに、[MODE][5]と押して指定するラジアンモード("RAD"点灯)、[MODE][6]と押して指定するグレードモード("GRA"点灯)があります。この角度単位は三角関数などを使うときに必要となります。電源スイッチONでは"DEG"が表示されます。
状態表示はほかにも[MODE][1]と押すプログラム書き込みモード("WRT"点灯)、[MODE][2]と押すトレースモード("TR"点灯、41ページ参照)、[MODE][7]と押すプリントモード("PRT"点灯、145ページ参照)、[MODE][9]と押す電子メモ入力モード("[MEMO][IN]"点灯)、[MODE][・]と押す拡張モード("EXT"点灯)を示すものがあります。

このような状態表示はさわっているうちに覚えてきますので、ここでは見るだけにしてもかまいません。

では、計算機にさわって表示をさせてみましょう。
もしさわっているうちに状態表示が色々と点灯してしまった時は、一度電源スイッチを切ってから再び電源を入れてください。

まず初めに簡単な計算をしてみましょう。

例） 123+456=579

_

ACを押します。
数式の通りにキーを押します。
1 2 3 + 4 5 6

123+456_

この後に =のかわりにEXEで答を求めます。
EXE

579

どうでしたか、計算が簡単にできますね。

では、次にもう少し長い計算をしてみましょう。

例） 45×6+89=359

ここでは45をまちがえて46と押してしまったとします。
4 6 * 6 + 8 9

46*6+89_

45を46と押してしまったことに気がつきました。でも、あわてずに、カーソルキー(⇦)でカーソルをまちがった所に合わせます。
⇦ ⇦ ⇦ ⇦ ⇦ ⇦

46*6+89

カーソルと6とが交互に点滅します

ここで正しい5のキーを押します。

5

45*6+89

これで計算式が正しくなりましたので、答を求めます。
EXE

359

このように、途中でまちがいに気付いたときは、カーソルキーを使って簡単に訂正することができます。ただし、EXEキーを押してしまった後では、最初から入れなおしてください。

それでは、アルファベットキーを使って文字を書いてみましょう。
アルファベットキーはタイプライターと同じ配列になっており、この配列をASCII型配列といいます。現在のポケコンはこのASCII型配列のキーボードが主流ですので、最初は慣れないでしょうが、だいたいの位置は覚えておいてください。

まず、文字を書いてみましょう。

例）文字は「ABCXYZ」とします。

ABCと書いてみます。

[A][B][C]　　　ABC_

次にXYZと書いてみます。

[X][Y][Z]　　　ABCXYZ_

それでは、ABCとXYZの間を1文字分あけてみます。
カーソルをXに合わせます。

[←][←][←]　　　ABCXYZ

1文字分あけます。

[SHIFT][INS/DEL]　　　ABC_XYZ

文字間をあけるときは、あけたい箇所の次にカーソルを合わせ、[SHIFT][INS/DEL]で1文字分あきますので、何文字分もあけたいときは、繰り返し同じ操作をします。

本機には数字・アルファベットのほかに、いくつかの特殊文字と呼ばれる文字があります。これはゲームや科学記号に便利な文字です。特殊文字の種類については4ページを参照してください。

ここで少し表示させてみましょう。

例）♤♡◇♧のマークを表示させる。

まず拡張モードを指定します。

[AC][MODE][・]　　　点灯します EXT _

このマークは[SHIFT]キーに続けてアルファベットキーを押します。

[SHIFT][H][SHIFT][J][SHIFT][K][SHIFT][L]　　　♠♥♦♣_

例）ΣΩμの記号を表示させる。

拡張モードになっているので、そのまま[SHIFT]キーに続けて押します。

[SHIFT][B][SHIFT][N][SHIFT][M]　　　♠♥♦♣ΣΩμ_

このようなマークや記号が用意されていますので、色々と利用して使ってみてください。なお、拡張モードになっているときに、今迄のアルファベット大文字を表示するモードに戻す場合は、もう一度MODE・と押し、"EXT"を消します。これでキーの押し方はだいたいわかったかと思います。このように色々とさわっているうちに"ERR 2"が表示されて、キーを押しても動かなくなることがあります。これは故障ではなく、まちがった命令をしましたというメッセージで、**「エラーメッセージ」**と呼ばれるものです。このときはあわてずに、ACキーを押せば表示が消えて、また動くようになります。このようなエラーメッセージにはいくつかありますので、詳しくは36ページまたは164ページをご覧ください。

2−2 データバンクはとても便利

本機の特長としてデータバンク機能があります。このデータバンク機能はMEMOキーを使うだけで簡単にメモデータを記憶させたり呼び戻したりすることができ、少し応用を加えるだけで色々な使い方があります。

たとえば　電話帳
　　　　　時刻表
　　　　　スケジュール表
　　　　　各種早見表　　など……

また、BASICプログラムの中から検索、呼び出し、書き込みもできますので、さらに利用範囲を広げることができます。

たとえば　得意先リスト
　　　　　製品リスト
　　　　　見積計算
　　　　　図書文献メモ
　　　　　ゴルフ集計　　など……

もっと色々な使い方もあると思います。
このデータバンクの使い方や応用例につきましては、別冊の「データバンク活用法」をご覧ください。

2−3 まず始めに基本計算

ここでは簡単な四則計算ぐらいを行なってみますが、関数電卓を使ったことのない方は注意していただきたいのです。本機は完全数式通りというたし算、ひき算よりかけ算、わり算を先に計算する機能を持っているからです。

例1)　23+4.5−53=−25.5

操作)　23 [+] 4.5 [−] 53 [EXE]　　　-25.5

※ここからは、数字キーはワクをはずして記します。

例2)　56×(−12)÷(−2.5)=268.8

操作)　56 [*] [−] 12 [/] [−] 2.5 [EXE]　　　268.8

※負符号がつく場合は、数字の前に[−]キーを押します。

例3)　7×8−4×5=36

操作)　7 [*] 8 [−] 4 [*] 5 [EXE]　　　36

※かけ算を先に計算してから、ひき算を計算します。

例4)　$(4.5\times10^{75})\times(-2.3\times10^{-78})=-0.01035$

操作)　4.5 [E] 75 [*] [−] 2.3 [E] [−] 78 [EXE]　　　-0.01035

※指数部は[E]キーに続いて押します。

もう一つの計算として、メモリーを使った代数計算があります。この計算は、ある一定の数値を色々と計算するときに便利です。

例えば　$3x+5=$

$4x+6=$

$5x+7=$

というような計算があり、xの値が123.456であるとき、同じ数値を繰り返し押すのは面倒なものです。この計算を手間をかけずに行なう方法はないでしょうか。解決策は変数と呼ばれるメモリーを使うことです。この例では代数計算にxという代数を使っていますので、変数Xを使って計算します。

まず、変数Xに123.456を入れます。

X = 123.456 EXE

この = は等しいという意味ではなく、変数Xに123.456を入れるという意味です。
では、計算をしてみましょう。

操作	表示
3 * X + 5 EXE	375.368
4 * X + 6 EXE	499.824
5 * X + 7 EXE	624.28

こんなに簡単にできます。

本機にはこのような変数がA～Zまで26個ありますので、色々な数値を覚えておくことができます。
この例では変数Xの数値は一定で、計算式が異なりますが、逆に計算式が一定で、変数の値が異なる場合はどうでしょう。
もし、計算式が"3 x + 5 ="と決っていて、xの値が 123、456、789 と変化する場合、今のような方法では操作が面倒になります。実際には計算式を計算機が覚えてくれて、変数Xの値だけを変えればよいのです。この便利な計算方法を**「プログラム計算」**といいます。本機はこのプログラム計算が得意なのです。ここではプログラムを使う前のマニュアル計算を行なっていますので、プログラムについては、第3章のプログラム編をご覧ください。

2－4 関数計算もおてのもの

本機は一般の四則計算のほかに、関数計算の機能も持っています。
この関数機能はプログラム中に組み込んでも使えますが、ここではマニュアルでの使い方について説明します。

本機に組み込まれている関数は次の通りです。

関数名		書 式	引数範囲
三角関数	sin x	SIN x	度：$\|x\|<1440$
	cos x	COS x	ラジアン：$\|x\|<8\pi$
	tan x	TAN x	グレード：$\|x\|<1600$
逆三角関数	$\sin^{-1} x$	ASN x	$\|x\|\leqq 1$
	$\cos^{-1} x$	ACS x	$\|x\|\leqq 1$
	$\tan^{-1} x$	ATN x	
平方根	$\sqrt{x}$	SQR x	$x\geqq 0$
常用対数	log x	LOG x	$x>0$
自然対数	ln x	LN x	$x>0$
指数関数	e^x	EXP x	$-10^{10}<x\leqq 230.2585092$
べき乗	x^y	x↑y	$x<0$のとき、yは自然数
10進→60進		DMS$(x)※	$\|x\|<10^{100}$、但し変換は$\|x\|<100000$
60進→10進		DEG(x, y, z)※	$\|$DEG$(x, y, z)\|<10^{100}$
整数化		INT x	
整数部除去		FRAC x	
絶対値化	$\|x\|$	ABS x	
符号化	(正数→1 0→0 負数→-1)	SGN x	
四捨五入	(xの10^yを 四捨五入)	RND(x, y)※	$\|y\|<100$
乱数		RAN#	

※DMS$、DEG、RNDは引数を必ず(　)でくくります。

では、関数を使って計算をしてみましょう。

● **三角関数**(sin、cos、tan)、**逆三角関数**(sin^{-1}、cos^{-1}、tan^{-1})

三角・逆三角関数を使うときは、必ず角度単位の指定(DEG、RAD、GRA)を行なってください。(角度単位を変更しない場合は、新たに行なう必要はありません。)

〈例〉 sin 12.3456°=0.2138079201

〈操作〉 MODE 4 →"DEG"

SIN 12.3456 EXE　　0.2138079201

※ここからは、アルファベットキー、数字キーはワクをはずして記します。

〈例〉 cos 63°52′41″=0.4402830847

〈操作〉 COS DEG SHIFT (63 , 52 , 41 SHIFT) EXE

0.4402830847

〈例〉 2・sin45°×cos65.1°=0.5954345575

〈操作〉 2 * SIN 45 * COS 65.1 EXE　　0.5954345575

〈例〉 $sin^{-1}0.5=30°$

〈操作〉 ASN 0.5 EXE　　30

〈例〉 cos ($\frac{\pi}{3}$rad) =0.5

〈操作〉 MODE 5 →"RAD"

COS SHIFT (SHIFT π / 3 SHIFT) EXE　　0.5

〈例〉 $cos^{-1}\frac{\sqrt{2}}{2}$=0.7853981634rad

〈操作〉 ACS SHIFT (SQR 2 / 2 SHIFT) EXE　　0.7853981634

〈例〉 tan(−35gra)=−0.612800788

〈操作〉 MODE 6 →"GRA"

TAN − 35 EXE　　-0.612800788

● **対数関数**(log、ln)、**指数関数**(e^x、x^y)

〈例〉 log 1.23(=log_{10}1.23)=0.0899051114

〈操作〉 LOG 1.23 EXE　　0.0899051114

〈例〉 ln 90($=\log_e 90$)=4.49980967

〈操作〉 LN90 EXE 　　4.49980967

〈例〉 e^5=148.4131591

〈操作〉 EXP5 EXE 　　148.4131591

〈例〉 $10^{1.23}$=16.98243652

（常用対数1.23の真数を求める）

〈操作〉 10 SHIFT ^ 1.23 EXE 　　16.98243652

〈例〉 $5.6^{2.3}$=52.58143837

〈操作〉 5.6 SHIFT ^ 2.3 EXE 　　52.58143837

〈例〉 $123^{\frac{1}{7}}$($=\sqrt[7]{123}$)=1.988647795

〈操作〉 123 SHIFT ^ SHIFT (1 / 7 SHIFT) EXE 　　1.988647795

〈例〉 log sin40°+log cos35°=−0.278567983

その真数は……0.5265407845(sin40°×cos35°の対数計算)

〈操作〉 MODE 4 →"DEG"

LOG SIN 40 + LOG COS 35 EXE 　　-0.278567983

10 SHIFT ^ SHIFT ANS EXE 　　0.5265407845

● その他の関数(√、SGN、RAN#、RND、ABS、INT、FRAC)

〈例〉 $\sqrt{2}+\sqrt{5}$=3.65028154

〈操作〉 SQR 2 + SQR 5 EXE 　　3.65028154

〈例〉 正数であれば"1"を、負数であれば"−1"を、0であれば"0"を与える。

〈操作〉 SGN 6 EXE 　　1

SGN 0 EXE 　　0

SGN − 2 EXE 　　-1

〈例〉 乱数発生(0＜RAN#＜1の擬似乱数)

〈操作〉 RAN SHIFT # EXE 　　0.7903739076

〈例〉 12.3×4.56の答を10^{-2}の位で四捨五入する。
12.3×4.56=56.088

〈操作〉 RND SHIFT ␣ 12.3 * 4.56 , − 2 SHIFT ␣ EXE　　56.1

※RND(x , y)のときのyは$|y|<100$

〈例〉 |−78.9÷5.6|=14.08928571

〈操作〉 ABS SHIFT ␣ − 78.9 / 5.6 SHIFT ␣ EXE　　14.08928571

〈例〉 $\frac{7800}{96}$の整数部は……81

〈操作〉 INT SHIFT ␣ 7800 / 96 SHIFT ␣ EXE　　81

※この関数は元の数値をこえない最大の整数を求めます。

〈例〉 $\frac{7800}{96}$の小数部は……0.25

〈操作〉 FRAC SHIFT ␣ 7800 / 96 SHIFT ␣ EXE　　0.25

●有効桁数指定、小数以下指定

有効桁数と小数以下の指定は"SET"コマンドにより行ないます。

有効桁数指定……SET E n (n=0~8)
小数以下指定……SET F n (n=0~9)
指定解除…………SET N

※マニュアル計算での有効桁数指定の場合の"SET E 0"は8桁指定となります。

※指定を行なうと、指定桁の下1桁目を四捨五入して表示します。
なお、計算機内部やメモリー内にはもとの数値が残っています。

〈例〉 100÷6=16.66666666………

〈操作〉 SET E4 EXE (有効桁数4桁指定)
100 / 6 EXE　　1.667 E01

〈例〉 123÷7=17.57142857……

〈操作〉 SET F2 EXE (小数以下2桁指定)
123 / 7 EXE　　17.57

〈例〉 1÷3=0.3333333333………

〈操作〉 SET N EXE (指定解除)
1 / 3 EXE　　0.3333333333

● 10進↔60進変換(DEG、DMS $)

〈例〉 14°25′36″=14.42666667

〈操作〉 DEG SHIFT (14 , 25 , 36 SHIFT) EXE

14.42666667

〈例〉 12.3456°=12°20′44.16″

〈操作〉 DMS SHIFT $ SHIFT (12.3456 SHIFT) EXE

12° 20′ 44.16

〈例〉 sin 63°52′41″=0.897859012

〈操作〉 MODE 4

SIN DEG SHIFT (63 , 52 , 41 SHIFT) EXE

0.897859012

関数計算もこのように簡単にできます。

第3章

BASICプログラムとは

この章では、BASICプログラムというものがどういうものかを見ていただき、どのように使うかを説明していきます。プログラムに使われるコマンドや関数については次の4章および5章でくわしく説明していきます。

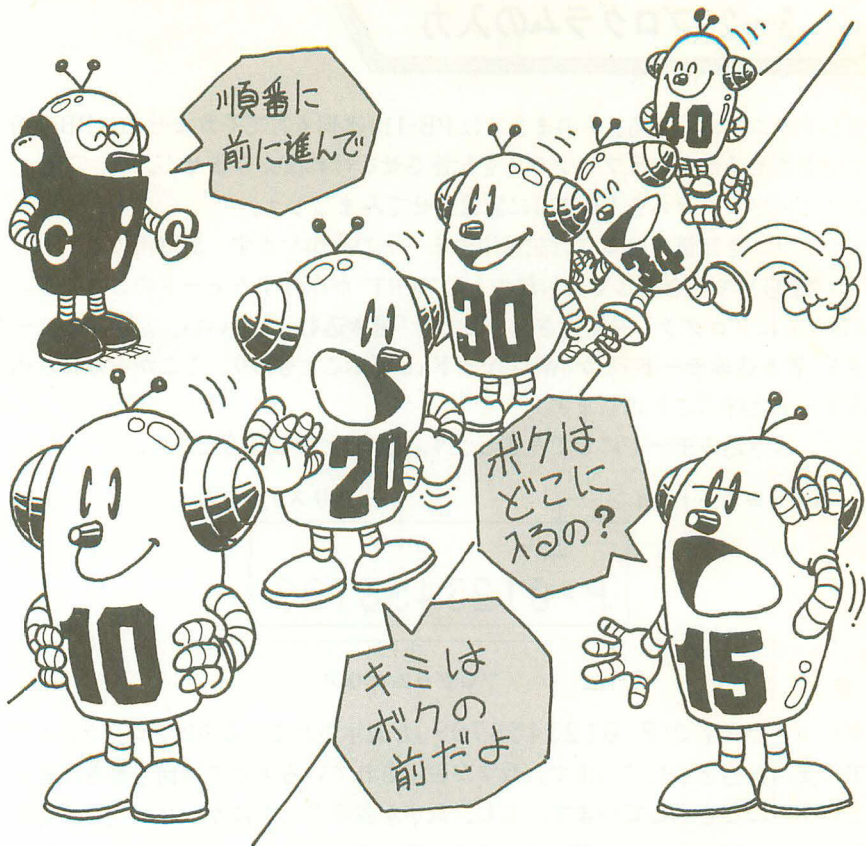
3−1 プログラムとは?

「プログラム」と聞くとよく難しいものだと感じる方がいますが、プログラムにも簡単なものから難しいものまであり、楽しく遊べるゲームにしても、立派なプログラムです。
ここでは難しいことはぬきにして、ゲームで遊びながらプログラムについて見ていきましょう。
まず、プログラムとはどういうものかを見てみましょう。

```
10 A=INT(RAN#*10)
20 INPUT B
30 IF A≠B THEN 20
40 PRINT "GOOD"
50 END
```

このプログラムは乱数により0から9までの1桁の数を作り出し、その数を当てると「GOOD」と表示されます。
ここに出てくるプログラム用の命令や関数は後で説明しますので、ここではプログラムとはどういう形をしているかを見てみます。
最初の行は、

```
10 A=INT(RAN#*10)
┬
行番号
```

頭に10という番号がついています。この番号のことを**「行番号」**と呼び、プログラムの実行順を示します。行番号は1から9999の間(1≦行番号<10000)であれば、いくつでもかまいませんが、番号の小さい方から順に実行していきますので、何でもよいというわけにはいきません。この例のように行番号を10、20、30……と10から10刻みにしているのは、プログラムを作っていくうちに後で途中に追加したくなることがあります。このようなときにもし1、2、3……となっていれば、追加ができなくなります。このために通常は10刻みで間をあけて行番号をつけます。

行番号の次に続いているのが、PB-110に仕事をさせるための命令です。

「A＝INT(RAN＃＊10)」というのは、乱数発生関数(RAN＃：54ページ参照)により0から1の間(0＜乱数＜1)の小数点以下10桁のでたらめな数を作り、その数を10倍して整数部分を取り出すと0から9までの数ができます。この数をAという変数(数値を記憶する箱)に入れなさいという命令です。

このように、プログラムとは実行の順番を示す行番号と、仕事を示す命令とがいっしょになってできています。

プログラムに使われている命令については第4章の実践プログラム編を参照してください。

3－2 プログラムの入力

プログラムがあっても、そのままではPB-110は何もしてくれません。PB-110に何をさせるかというプログラムを記憶させなければなりません。
ここではプログラムをPB-110に記憶させてみましょう。
プログラムを記憶させるには記憶用のモードで操作します。記憶用のモードとは、[MODE][1]とキーを押して表示部の上に"WRT"が点灯するモードのことです。
PB-110にプログラムを記憶させることを**「書き込む」**ともいい、記憶用のモードを**「書き込みモード」**とか**「WRTモード」**と呼ぶこともあり、ここからは書き込みモードと呼ぶことにします。
では、書き込みモードにしてみましょう。[MODE][1]と押してください。

下の大きな文字で「P 0123456789」と表示されているのはプログラムエリアの使用状態を示しています。数字が表示されているエリアは何も書き込まれていないことを示しています。もし、数字が表示されずにカーソル(_)が表示されていれば、そのプログラムエリアにプログラムが入っていることを示しています。
表示窓の上部に表示している小さな数字は残りステップ数を示しています。残りステップ数は増設RAMパック〈OR-1Ⓔ〉を装着していないときに、プログラムおよびデータバンクに何も記憶していない状態で544ステップです。
ここでは、全部のプログラムを消すために、次のように操作します。

NEW ALL [EXE]

この操作は全プログラムと変数の内容を消して、プログラムエリアP0にプログラムを記憶させる準備にもなります。
では、前に出てきたプログラムを記憶させてみましょう。

[1][0][A][=][I][N][T][SHIFT][(][R][A][N][SHIFT][#][*][1][0][SHIFT][)] [EXE]
↑一行ごとの最後に必ず押します。

[2][0][SHIFT][INPUT][B] [EXE]
└アルファベットキーを1つづつ押して[I][N][P][U][T]としても同じです。

[3][0][SHIFT][IF][A][SHIFT][≠][B][SHIFT][THEN][2][0] [EXE]

[4][0][SHIFT][PRINT][SHIFT]["][G][O][O][D][SHIFT]["] [EXE]

[5][0][E][N][D] [EXE]

以上のキー操作でプログラムが記憶されました。
もし、うまくできなくてもあわてずに、ゆっくりと確実にキーを押してください。まちがえても、次のように操作すれば簡単に訂正できます。

■ EXE キーを押すまえにまちがいに気付いた

このときは、EXE キーを押さずに ⇦⇨ キーを使ってまちがえた箇所にカーソルを合わせ、訂正します。

例1) 「B」と押すところを「V」と押してしまった。

```
20 INPUT V_
```

● ⇦ キーを1回押して「V」に合わせる。

⇦

```
20 INPUT V
```

● 続けて正しいキーを押して EXE キーを押します。

B EXE

```
20 INPUT B
```

例2) 行番号10の「＊」を抜かしてしまった。

```
INT(RAN#10)_
```

● ⇦ キーを使って入れたい箇所の次にカーソルを合わせます。

⇦⇦⇦

```
0A=INT(RAN#1
```

● 1文字分あけます。

SHIFT INS

```
0A=INT(RAN#_
```

●「＊」を入れます。

＊

```
A=INT(RAN#*1
```

● 訂正が終ったら EXE キーを押します。

EXE

```
10 A= INT( R
```

例3) 行番号40の「GOOD」のOを1文字多く入れてしまった。

```
INT "GOOOD"_
```

● ⇦ キーを使って「O」に合わせます。

⇦⇦⇦

```
PRINT "GOOO
```

● 1文字分削除しますので、DEL キーを押します。

DEL

```
PRINT "GOOD
```

●削除が終りましたら EXE キーを押します。

EXE　　　40 PRINT "GO

■ EXE キーを押してからまちがいに気付いた

EXE キーを押した後ではプログラムとして記憶されてしまいますので、LISTコマンドを使って再び呼び出し、正しく訂正します。

例）行番号30の「A≠B」を「A＝B」とまちがえてしまった。

●LISTコマンドで行番号30を呼び出します。

SHIFT LIST 30 EXE　　　30 IF A=B TH

└ L I S T と押しても同じです。

●⇦キーでまちがえた「＝」に合わせます。

⇦⇦⇦⇦　　　30 IF A=B TH

●正しくキー操作します。

SHIFT ≠　　　30 IF A≠B TH

●訂正が終りましたら、必ず EXE キーを押します。

EXE　　　40 PRINT "GO

●もし行番号40が記憶されていれば、行番号40のプログラムが表示されます。このほかに訂正がなければ AC キーを押して訂正を終らせます。

AC　　　_

EXE キーを押した後でも、このようにLISTコマンドで呼び出して訂正することができます。なお、新たに行番号をつけて書き直すこともできます。すでにある行番号をつけて記憶させた場合は、後から記憶させた方が優先され、前に記憶させた行は消えます。

このようにしてプログラムを記憶させますが、記憶の操作が終了したら、MODE 0 と押してRUNモードにします。書き込みモードのままでは記憶させたプログラムを消したり、書きかえてしまったりすることがありますので、記憶の操作終了後は必ずRUNモードにしてください。

3−3 プログラムの実行

では、前に記憶させたプログラムを実行させてみましょう。
MODE 0 と押してRUNモード(RUN点灯)であることを確認してください。
プログラムを実行させるには2つの方法があります。

①プログラムエリア指定による実行
SHIFT キーに続いて 0 〜 9 の数字キーを押しますと、プログラムエリアの指定となり、そのプログラムエリアにプログラムが記憶されていれば、プログラムがスタートします。

例) SHIFT P0

②RUNコマンドによる実行
プログラム実行開始コマンドRUNコマンドにより実行させます。

例) SHIFT RUN (R U N としても同じ) EXE

この2つの実行方法の違いは、①の方法では必ずプログラムエリアの先頭からスタートしますが、②の方法では先頭からも、任意の行番号からもスタートすることができます。

例) SHIFT RUN 1 0 0 EXE →行番号100から実行開始

では、RUN EXE

?

カンを働かせて

5 EXE

?

はずれです。もう1度

9 EXE

?

またまたはずれ。もう1度

1 EXE

?

⋮

2 EXE

GOOD　　STOP

PRINT文を実行すると"STOP"が点灯します。

やっと当りました。
(乱数を使っているため、正解は2とは限りませんが)

プログラムを作り上げ、PB-110に記憶させた後、すぐにでも実行させてみたいものです。もし、実行させてエラー(ERR表示)になっても、がっかりしないでください。こんなときには次の項目を読んでまちがい探し(デバッグと言う)を行なってください。

3―4 デバッグ(まちがいを直す)

プログラムを作り上げ、いざ実行開始というときに、実行させてもエラーが表示されたり、結果が思うように得られない。このような事はよくあることですので、がっかりせずに、原因を追求してください。
プログラム中にある「**まちがい**」のことを「**バグ**」(虫の意味)と呼び、この虫を取り除くという意味で「**デバッグ**」といいます。
このデバッグの方法も原因により異なります。実行中にエラー表示になる場合と、エラーにはならないが結果が思うように得られない場合です。
実行中にエラーが表示される場合は、PB-110がエラーの発生した箇所とエラーの内容を知らせてくれますので、原因追求は比較的簡単にできますが、エラーは表示しないが結果が思うように得られない場合は、けっこう厄介なものです。
では、順を追ってデバッグをしてみましょう。

(1) エラー表示によるデバッグ

エラー表示とは、**エラーメッセージ**とも呼ばれ、

このように、3つの要素をおしえてくれます。
エラー内容は、〝ERR〟に続くコードナンバーにより、どのようなエラーが起きたかを知らせてくれます。このコードナンバーは1から9の数字で、〝ERR1〟は「メモリーオーバー」とか、〝ERR2〟は「構文エラー」というような内容が決っています。このコードナンバーの内容については、巻末164ページの「エラーメッセージ一覧表」を参照してください。
エラー発生プログラムエリアは、エラーの発生したプログラムエリアを知らせてくれます。
エラー発生行番号は、エラーの発生した行番号を知らせてくれます。
この3つの要素により「**どこで**」、「**どのような**」エラーが起きたのかを知ることができます。
それでは例を上げ説明していきましょう。
よく起きるエラーに〝ERR2〟があります。これは「**構文エラー**」といい、記憶させたプログラムに誤りがあると起きるのです。

前の例題で使ったプログラムをまちがえて記憶させてみます。

操　作	表　示
MODE 1	P _123456789
SHIFT LIST EXE	10 A= INT(R
⇨⇨⇨⇨ DEL	= INT(RAN#1
EXE	20 INPUT B_
AC MODE 0	READY P0

これは行番号10の「RAN＃＊10」を「RAN＃10」とまちがえてみました。
では実行させましょう。

操　作	表　示
SHIFT RUN EXE	ERR2 P0-10

エラーメッセージが表示されました。これはプログラムエリアP0の行番号10で構文エラーが発生しましたという意味です。
では行番号10を調べてみましょう。

操　作	表　示
AC………エラー解除	_
MODE 1	P _123456789
SHIFT LIST EXE	10 A= INT(R

⇨キーを押して左から順に調べていきます。

操　作	表　示
⇨⇨⇨⇨	= INT(RAN#1

正しいプログラムと合っているかチェックしていきます。
「＊」が抜けていることがわかりましたので、正しく直します。

操　作	表　示
SHIFT INS	= INT(RAN#_
＊	INT(RAN#*1
EXE	20 PRINT B_
AC MODE 0	READY P0

このように、〝ERR2〞はプログラムの入力ミスが多いので、〝ERR2〞が表示されたときはエラーの発生した行番号のプログラムをよくチェックしてください。
なお、記憶のまちがいだけでなく、READ文(78ページ参照)でデータを読み込むときに、数値変数に文字データを読み込もうとすると〝ERR2〞が表示されます。〝ERR2〞の発生した場所にREAD文があるときは、DATA文中のデータもチェックしてください。
では、エラーの種類別にチェックポイントを上げてみましょう。

●ERR 1：メモリー不足。スタックオーバー。

残りステップ数を確認して、DEFM文でステップ数以上をメモリーに変換しようとしたかチェックする。

●ERR 2：構文エラー

記憶させたプログラムにまちがいがないかチェックする。

●ERR3：数学的エラー

数式の演算結果が10^{100}以上であったり、関数の入力範囲をこえていないかをチェックする。特に変数を使っている場合は、前後関係から変数の内容をチェックする。(0による除算や、負数の平方根をとっている場合が多い。)

●ERR4：未定義行番号エラー

GOTO文やGOSUB文、RESTORE文による行番号の指定が正しくないので、行番号を確認する。

●ERR5：引数エラー

引数やパラメータを必要とするコマンドや関数において、引数やパラメータの値をチェックする。特に変数を使っている場合は、前後関係から変数の内容をチェックする。

●ERR6：変数エラー

配列変数を使うときに、DEFM文でメモリーの増設を行なっているかチェックする。また、同じメモリーを文字変数と数値変数の両方に、同時に使っていないかをチェックする。

●ERR7：ネスティングエラー

エラーの起きた行がRETURN文やNEXT文であれば、正しくGOSUB文やFOR文に対応しているかチェックする。また、エラーの起きた行がGOSUB文やFOR文であれば、ネスティングのくり返しをチェックして、GOSUB文は8回まで、FOR文は4回までとする。

●ERR8：パスワードエラー

パスワードが設定されているときに、別のパスワードを入力しようとしたり、使えないLISTコマンドや、NEW、NEW ALL等の操作をしたかチェックする。

●ERR9：オプションエラー

カセットインタフェイス<FA-3>やミニプリンタ<FP-12>または<FP-12S>が正しく接続されているかチェックする。<FP-12>または<FP-12S>は充電されているか、または紙づまりをしていないかチェックする。<FA-3>に接続しているテープレコーダーの音量調整やトーンの調整を変えて再び行なったり、テープレコーダーのヘッドを掃除したり、テープを新しいものと変えてみる。録音のときは白プラグのみを、再生のときは黒プラグのみを差し込んで試してみる。

⑵エラーは表示されないが、結果が思うようにならない。

このような場合のデバッグは、プログラム中の計算式や変数の与え方や、IF文による判断がまちがえていたりすることが多いので、計算式や変数、IF文による判断を重点的にチェックしてください。
では前の例題を使ってまちがえてみましょう。

操　作	表　示
MODE 1	P _123456789
SHIFT LIST 30 EXE	30 IF A≠B TH
⇦⇦⇦⇦	30 IF A≠B TH
=	30 IF A=B TH
EXE	40 PRINT "GO
AC MODE 0	READY P0

これでは答えがまちがっていたら「GOOD」を表示します。
では実行してみましょう。

操　作	表　示
SHIFT RUN EXE	?
5 EXE	GOOD

正解である「GOOD」が表示されました。しかし、確認のため答えを見てみましょう。

操　作	表　示
A EXE	2

答えは「２」です。プログラムが正しく働いていません。
プログラムの流れを見るために、答えを作る行番号10の後にSTOP文を入れてみます。STOP文はプログラムの実行を一時停止させる命令です。

操　作	表　示
MODE 1	P _123456789
1 5 S T O P EXE	15 STOP
MODE 0	READY P0

では実行をしてみます。

操　作	表　示
SHIFT RUN EXE	_ （STOP ── STOP文で停止中点灯）
A EXE（答えを見る）	9
EXE　（実行を続ける）	?
9 EXE（答えを入力）	?

正しい答えを入れたにもかかわらず「GOOD」は表示されません。これは、判断が正しく働いていないことを示しています。
IF文の入っている行番号30を調べてみます。

操　作	表　示
MODE 1	P _123456789
SHIFT LIST 30 EXE	30 IF A=B TH

ここで、行番号30をよく調べます。「≠」と「＝」がまちがっていたことがわかります。
それでは、正しく直してみましょう。

操　作(続けて)	表　示
⇦⇦⇦⇦	30 IF A=B TH
SHIFT ≠	30 IF A≠B TH
EXE	40 PRINT "GO
AC 1 5 EXE（不要な行番号15を削除）	_
AC MODE 0	READY P0

これでデバッグは完了です。

このSTOP文によるデバッグの他に、トレースモード(MODE②と押す。TR点灯)によるデバッグがあります。

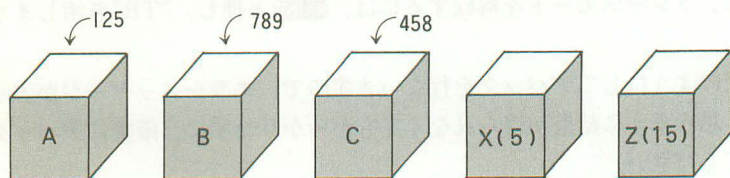
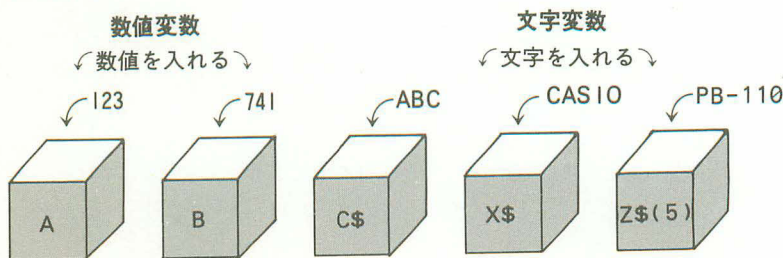
トレースモードでのデバッグは、プログラムを1命令ごとに実行して停止します。この停止している間に変数の値を見たりできますので、EXEキーで1命令ごとに進めてデバッグをします。

この例でためしてください。EXEキーを押すごとにプログラムエリアと行番号を表示します。

なお、トレースモードを解除するには、MODE③と押し、"TR"を消します。

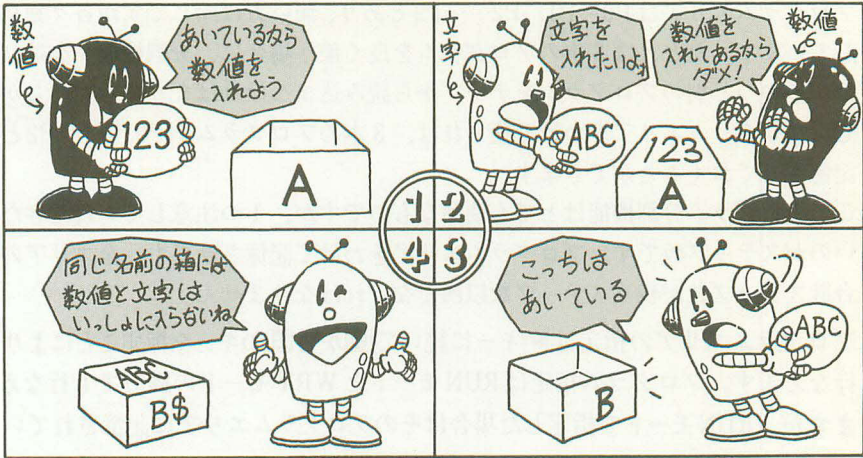
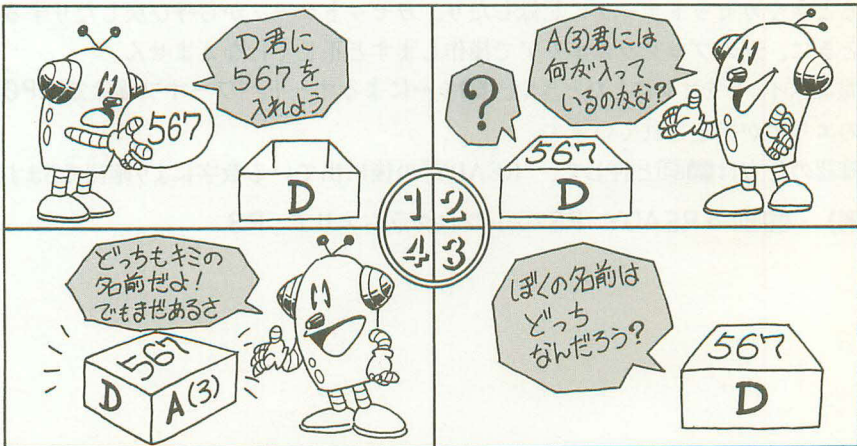
以上のようにしてデバッグを行ないますので、エラーメッセージが表示されたり、思うように結果が得られなくてもがっかりせずに、確実にデバッグを行なってください。

3－5 変数とは

プログラムを作る上で重要な要素として、変数があります。変数とは、入力したデータや計算の答を入れておく箱で、各々に名前がついています。この変数には標準のAからZまでの変数と、配列変数と呼ばれAからZの名前に区別をする添字のついたA(5),B(50)という変数があります。

この変数にはさらに2種類あり、数値を入れる数値変数と文字列を入れる文字変数があります。今迄でできた変数は計算をするための数値を入れておくので、数値変数でしたが、この他にAからZの変数名に$(ドルマーク)をつけたA$,B$,C$と書いて使う文字変数があります。PB-110にはこの他に、専用文字変数($)という特別な文字変数があります。
数値変数には10桁(仮数部10桁、指数部2桁)までの数値が入り、文字変数には7文字までの文字列が入ります。また、専用文字変数には30文字までの文字列が入ります。

この2種類の変数は入れるものが異なりますので、数値変数に"ABC"のような文字を入れることはできませんし、文字変数に数値を入れることもできません。
これ等の変数は用途により使い分け、計算のための数値を入れるのであれば数値変数を使い、メッセージや記号を入れるのであれば文字変数を使います。
配列変数は、多くの変数を使いデータを記憶していくときに便利で、1番目、2番目……というように順番で示される番号で変数を区別します。配列変数については、プログラム中で使いながら説明していきますので、ここでは変数の種類として覚えておいてください。

〈変数使用上の注意〉

PB-110では、数値変数と文字変数は変数名が同じであれば同じ箱を使っています。

このため、1つのプログラム内でAという数値変数とA$という文字変数を同時には使えません。

また、配列変数を使うときも、同じ箱を色々な名前で呼ぶことがありますので、重ねて使わないようにしてください。

同じ箱でも、数値変数「D」として呼ぶときもあれば、配列変数「A(3)」として呼ぶときもあります。実際は1つの同じ箱ですが、用途に合わせて色々と名前がかわります。

変数名と配列変数との関係については、168ページの一覧表を参照してください。

3−6 プログラムエリア

本機の特長としてプログラム分割機能があります。これはプログラムエリアを10個にわけ、独立したプログラムを記憶させておくことができます。
プログラムエリアはP0、P1、P2……P9とあり、使い方は同じですが各々独立しています。たとえば3本のプログラムを良く使う場合に、分割機能がなければ、そのつど別のプログラムをテープから読み込まなければなりません。しかし、プログラムエリアをわけて使えれば、3本のプログラムをP0、P1、P2と記憶させておくことができます。
このプログラム分割機能はとても便利なものですが、1つ注意していただきたいのがステップ数です。プログラムエリアをわけて記憶させても、全エリアの合計ステップ数が残りステップ数以内でなければなりません。

プログラムエリアの指定はSHIFTキーに続いて0から9のキーを押すことにより行なえます。プログラム指定はRUNモードとWRTモードの両方でも行なえますが、RUNモードで指定した場合はそのプログラムエリアに記憶されているプログラムが自動的にスタートします。WRTモードで指定した場合はプログラムはスタートせずに、プログラムエリアの指定のみ切り替わります。
プログラムエリアの区別は、はっきりとしてください。プログラムを記憶させるときやカセットテープに記録したり、カセットテープから呼び戻したりするときに、別のプログラムエリアで操作しますと正しく行なえません。
電源スイッチをオンにしたときや、ACキーによるオートパワーオフ解除ではP0のエリアが指定されています。
確認の仕方はMODE0と押して、"READY"の後に出ている数字により確認できます。

例）　MODE0→READY　P3……プログラムエリア　P3

3-7 ステップ数のかぞえ方

PB-110の持っているプログラム容量は、標準で544ステップ、OR-1Ⓔ増設時で1568ステップです。
このステップとはプログラムを記憶できる許容量を示す単位で、プログラムを記憶させていくと、残りステップ数が減っていきます。
現在の残りステップ数は、MODE 1 と押してWRTモードにすると表示されます。

例） MODE 1

```
WRT          544  ――残りステップ数
P 0123456789
```

また、使用するステップ数は次のようにかぞえます。

- 行番号………… 1~9999のいくつであっても2ステップ
- コマンド……… 1ステップ
- 関数…………… 1ステップ
- 文字…………… 1文字で1ステップ
- 以上の他に、区切りとしてEXEキーを押して記憶させるのに1ステップ必要とします。

例）

```
1   INPUT  A  EXE  ………5ステップ
2   1      1  1
10  B=SIN  A  EXE  ………7ステップ
2   1 1 1  1  1
100 PRINT "B=" ;B EXE ………10ステップ
2   1      4   1 1 1
                   計22ステップ
```

- メモリーを増設した場合は、1つにつき8ステップ必要とします。

例） 標準状態……26メモリー、544ステップ
DEFM 10 EXE……36メモリー、464ステップ
※増設される変数はZ(1)~Z(10)までの10個です。

- パスワードを設定した場合は、パスワードの文字数+1ステップを必要とします。

例） PASS "CASIO" EXE
5 + 1 = 6ステップ

第４章

実践プログラム

ここでは簡単なプログラムを通して、コマンドや関数の使われ方を見てみましょう。コマンドや関数の文法上の規則や入力範囲については、次の５章を参照してください。
なお、ここからはプログラムリストを記載しますので、プログラムの入力(記憶)方法については、３章を参照してください。

4−1 数当てゲーム

このプログラムは3章で使ったものを、さらに1歩進めたものです。前の例題では入力した答えが正解と合っているかどうかしかわかりませんでしたが、このプログラムでは、入力した答えが正解より大きいか小さいかのメッセージを表示してくれます。

プログラムリスト

```
10 A=INT(RAN#*20+1):C=0
20 INPUT B
30 C=C+1
40 IF A=B THEN PRINT "GOOD";C:END
50 IF A<B THEN PRINT "HIGH";:GOTO 20
60 PRINT "LOW";
70 GOTO 20
```

（86ステップ）

操　作　例

まずはこのゲームを始めてみましょう。

RUN EXE　　`?`

1から20の間で答えを入力します。

15 EXE　　`HIGH?`

このメッセージは、入力した答えが正解より大きいことを示しています。

5 EXE　　`LOW?`

このメッセージは、入力した答えが正解より小さいことを示しています。

11 EXE　　`HIGH?`

まだまだ大きいようです。

9 EXE　　`GOOD 4`

「9」が正解でした。そして、正解を得るまでに4回かかりました。

ここで説明する命令および関数

INPUT，PRINT，IF～THEN，GOTO，END，INT，RAN#，
:(マルチステートメント)，代入文(LET)

■INPUT(データ入力)

INPUT文はプログラム中で必要なデータをキーボードから入力する命令です。INPUT文を実行すると、プログラムを一時停止し、「?」(クエスチョンマーク)を表示します。このときキーボードからデータ(数式や文字式)を入力し、EXEキーを押すとプログラムは続けて実行します。

```
20 INPUT B
```

この例のようにINPUTコマンドの次に変数を書き、指定された変数にデータが入ります。ここでは数値変数Bを使っていますので、データは1や2という数値または、DやEという数値変数、2＊3のような数式を入力します。数値変数に文字データを入力しようとするとエラー(ERR2)となり、ACキーを押した後、再入力ができます。
変数が文字変数の場合(INPUT Z$等)、文字型データが入力できます。
このINPUT文にはメッセージを表示させる働きがあり、入力内容を示すこともできます。

例)
```
 5 INPUT "LEVEL",L ←追加
10 A=INT(RAN#*L*10+1):C=0
                └─変更
20 INPUT B
   :
```

この例は行番号5を追加し、行番号10を少し変更していますので、行番号5で入力したレベルに合わせて、作られる答えの範囲が広がります。
このようにすれば、メッセージが表示されるため、とても便利になります。

表示 → LEVEL?

また、入力したいデータが複数個ある場合は、,(カンマ)で区切っていくつでも書くことができます。

例)
```
INPUT A,B$,C
INPUT "A=",A,"B$=",B$,"C=",C
```

この場合の入力方法は、1つづつEXEキーを押してデータを入力します。

例)
```
INPUT A,B$,C→ 10 EXE
               ABC EXE
               30 EXE
```

■PRINT(データ出力)

PRINT文は、データを表示する命令で、次に続くメッセージ(文字列)や変数の内容、計算式の結果を表示します。

例) PRINT "GOOD";C (行番号40)
PRINT "HIGH"; (行番号50)
PRINT "LOW"; (行番号60)

行番号40のように" "(ダブルクォーテーション)でかこまれた文字列はそのまま表示し、次のデータである変数Cの値を続けて表示します。最初のデータ"GOOD"と次のデータ変数Cの間にある;(セミコロン)は続けて表示するという意味を持ちます。

データの区切りには;の他に,(カンマ)が使えます。

この違いは、;が続けて表示するのに対し、,では,以前のデータを表示後、一時停止し、EXEキーが押されるのを待ちます。EXEキーが押されたなら、,以降のデータを表示します。

例) 10 INPUT "A",A,"B",B
20 PRINT "A=";A*2,"B=";B*2

この例は2つの数値を入力し、各々の2倍を求めるものです。
たとえば、5と7を入力してみます。

このように、;で区切った「"A=";A*2」と「"B=";B*2」は続けて表示されますが、この間の,により、この2つの表示は停止し、EXEキーを押すことにより次の表示に移ります。

行番号50と60のPRINT文の最後にある;は応用的な使い方で、この;がないと「HIGH」や「LOW」を表示後停止し、EXEキーを1度押してから次の答えを入力することになります。この手間をはぶくため、;を最後につけて表示が止まらないようにしています。表示が止まらないということは、次に行番号20へ戻り、「?」を続けて表示することになります。

このように、PRINT文は数式や文字列、数値変数や文字変数の内容を表示させます。

PRINTコマンドの応用としてよく使われるものに、データなしの「PRINT」だけがあります。

たとえば次のプログラムを実行してください。

```
例）　10 PRINT "TEST-1";
　　　20 PRINT "TEST-2"
```

このプログラムを実行すると、次のように続けて表示されます。

TEST-1TEST-2

これは行番号10の最後が；であるため、表示が停止せずに続けます。このとき、間に行番号15として「15 PRINT」を入れてみます。
すると表示は

TEST-2

これは行番号10が無視されているのではなく、一度「TEST－1」を表示後、すぐに表示を消し、次の「TEST－2」を表示しているのです。

このようにデータなしのPRINT文は、表示を消す働きを持っています。このような使い方はプログラムを停止せずに表示をかえるとき、よく使います。

■IF～THEN(判断する)

このIF～THEN文は、プログラム中で大小の比較をし、その結果が正しいかどうかで次の作業がかわるときに用いられます。

```
40 IF A=B THEN PRINT "GOOD";C:END
50 IF A<B THEN PRINT "HIGH";:GOTO 20
```

行番号40では作られた正解と入力された答えが合っているかを判断し、合っていれば「THEN」以降の文「PRINT "GOOD"；C：END」を実行します。もし合っていなければ正しくないので、次の行の行番号50に実行を移します。行番号50も同様に、正解より入力された答えの方が大きいときは「HIGH」を表示して行番号20へ移ります。
このように、IF～THEN文は「IF」と「THEN」の間の比較式が成り立つかどうかで作業を振り分けます。比較式に使える記号は次の6種類です。

＝：左辺と右辺が等しい。
≠：左辺と右辺が等しくない。
<：左辺より右辺が大きい。
>：左辺より右辺が小さい。
≦：左辺より右辺が大きいか、等しい。
≧：左辺より右辺が小さいか、等しい。

この6種類の比較により、正しいときはTHEN以降を実行します。
THEN以降にはこの例のような命令(PRINT文等)を書くことができますが、この他に分岐先を書くこともできます。

例) ① IF A=B THEN 100
　　② IF A<B THEN #9

①の使い方は「THEN GOTO 100」と同じ意味で、行番号100にジャンプ(作業の流れを移す)しなさいという意味です。②の使い方は「THEN GOTO #9」と同じ意味で、プログラムエリアP9にジャンプしなさいという意味です。
このように比較式が成り立てば「THEN」以降を実行するという大変便利な命令です。
この例で比較式に使っているのは数値変数AとBですが、この他に数値や計算式、文字変数や文字、文字式も使えます。

例) ① IF A>10 THEN……………数値変数と数値
　　　　Aの内容が10より大きければ正。
　　② IF B*2=C*3 THEN………計算式と計算式
　　　　B＊2の答えとC＊3の答えが等しければ正。
　　③ IF Y$="P" THEN…………文字変数と文字
　　　　Y$の内容が「P」であれば正。
　　④ IF N$>M$ THEN …………文字変数と文字変数
　　　　N$の内容がM$の内容より大きければ正。

この中の③や④のように文字の比較もできます。このときの大小関係は、アルファベットではA、B、C……Zの順に大きく、数字では0、1、2……9の順に大きくなっています。くわしくは、165ページのキャラクター表をご覧ください。
IF文の応用として複数の条件判断をすることがあります。

例) IF A>0 THEN IF A<10 THEN……

この使い方は、変数Aの内容が0より大きく、なおかつ10より小さいときだけ、最後のTHEN以降に進みます。つまり、ここでは変数Aの内容が1から9の間(1≦A≦9)のとき条件が成立するのです。
このようにして使えば、条件が3つでも4つでも1度に判断できますが、あまり長く重ねすぎるとプログラムが見づらくなったり、62文字に入らなくなりますので2つか3つぐらいがよいでしょう。

■GOTO（流れをかえる）

GOTO文は無条件ジャンプとも呼ばれ、プログラムの実行を次に示す行番号やプログラムエリアに移します。
行番号50では最後に「GOTO 20」と入っていますので、IF文が成立して「HIGH」を表示後、行番号20へジャンプします。もし、この「GOTO 20」が入っていなければ、次の行番号60へ進み「LOW」を表示してしまいます。
また行番号70ではプログラムの最後に「GOTO 20」と入っていますので、プログラムを終了せずに行番号20へとジャンプします。
プログラムは行番号の小さな順に実行していきますので、途中で次の行を実行したくないときや、始めの方の行に戻りたいときにGOTO文が使われます。
「GOTO」の後にはジャンプ先の行番号やプログラムエリア番号を書きます。

例） ① GOTO 50
　　 ② GOTO #8

①の使い方では行番号50にジャンプします。②の使い方ではプログラムエリアP8にジャンプし、P8に入っているプログラムの先頭から実行を続けます。
もし、該当する行番号がなかったり(ERR4)、P0～P9の範囲をこえた場合(ERR2)はエラーになります。

■END（プログラム終了）

END文はプログラムを終了させる命令で、プログラム中のどこに入れてもかまいませんが、この文が読まれた時点でプログラムを終らせます。
行番号40のような使われ方は、IF文により判断した結果、正しければ「GOOD」と回数（Cの値）を表示してプログラムを終了します。
通常プログラムの最後に入れて、後に続かない場合は省略してもかまいませんが、この例のように次にまだプログラムがあるときや、サブルーチン（66ページ参照）との区切りには必要となります。

■INT（元の数をこえない最大の整数を与える関数）

INT関数はインテジャーと呼ばれ、整数部を与える（取り出す）関数です。この関数はゲームで乱数から整数を作るときによく使われ、小数点以下の付いている数値から整数を取り出します。この取り出される整数は、元の値が正の値（0より大きい）のときは小数点以下を切りすてた形となりますが、元の値が負のとき（0より小さい）は少し違います。

例） INT 12.345 → 12
　　 INT−45.23 →−46

これは、INT関数が元の数をこえない最大の整数を求めるためで、負の場合は−46＜−45.23＜−45となり、答えは−46となるのです。

※このことをガウス関数(〔X〕)とも呼びます。

■RAN＃(乱数発生関数)

RAN＃関数は、0より大きく1より小さい(0＜乱数＜1)小数点以下10桁以内の乱数を与える関数です。

この関数は前記のINT関数と組み合わせてよく使われ、任意の数値を作り出す役目をします。

この例では1から20までの任意の数値を作るため、行番号10のように使われています。

```
10 A=INT(RAN#*20+1)
```

RAN＃関数で作られる値は小数点以下ですので、整数値が必要なときはこの例のようにします。

ではここで、いくつかの数値の求め方を見てみましょう。

例) ①0から9までの1桁の整数を求める。

```
INT(RAN#*10)
```

※最小 0.0000000001＊10＝0.000000001
　最大 0.9999999999＊10＝9.999999999
　ゆえに、0から9の整数が得られます。

②1から15までの整数を求める。

```
INT(RAN#*15+1)
```

RAN＃を15倍すると0～14の数が得られますので、これに1を加え、1～15の整数を求めます。

※最小 0.0000000001＊15＋1＝1.0000000015
　最大 0.9999999999＊15＋1＝15.9999999985
　ゆえに、1から15の整数が得られます。

③10から20までの整数を求める。

```
INT(RAN#*11+10)
```

RAN＃を11倍すると0～10の数が得られますので、10を加え10～20の整数を求めます。

※最小 0.0000000001＊11＋10＝10.0000000011
　最大 0.9999999999＊11＋10＝20.9999999989
　ゆえに、10から20の整数が得られます。

■マルチステートメント

マルチステートメントとは、文と文を：(コロン)でつなげて1行にする働きをもっています。

たとえば行番号10のような場合、

```
10 A=INT(RAN#*20          ➡   10 A=INT(RAN#*20+1)
   +1):C=0                     15 C=0
        (17ステップ)                    (19ステップ)
```

と2行に分けてもかまいませんが、1行にまとめた方がすっきりしますし、またステップ数も少なくてすみます。

行番号40や50での使い方は少し違います。この2つはIF文の中で使われ、THEN以降、すなわち条件が成立したときにだけ実行されるようになっているからです。もし、行番号40の「END」を次の行にすれば、行番号50以降は実行せずにプログラムが終了してしまいますし、行番号50の「GOTO 20」を次の行にすれば、行番号60を実行せずに、行番号20へ戻ってしまいます。

このように、マルチステートメントを使うとプログラムをまとめて、すっきりとすることができますが、あまり続けすぎると逆に見づらくなったりしますので、程々にしておく方がよいでしょう。

■代入文(LET)

代入文とは行番号10や30で使われているように、＝(イコール)の左辺にある変数に、右辺の式の結果を代入する(入れる)ことを言います。

この代入命令として「LET」という命令があります。この命令を使いますと、

例）
```
10 LET A=INT(RAN#*20+1):LET C=0
30 LET C=C+1
```

となります。ところが、48ページのように「LET」を書かなくても同じ意味になるのです。

実は「LET」という命令は命令自体が省略できますので、書かなくてもかまわないのです。通常はLETを省略して＝(イコール)の右辺の結果を左辺に代入することを代入文といいます。

4−2 モグラたたきゲーム

このプログラムは、表示窓の10個所の窓から出てくるモグラの位置をすばやく数え、該当する⓪から⑨の数字キーを押すゲームです。
モグラは一定時間顔を出していますが、まちがった窓のキーを押すと、逃げてしまいます。
モグラが10回顔を出すうち、あなたは何回たたけるでしょうか。

プログラムリスト

```
 10 N=0
 20 IF KEY$≠"" THEN 20
 30 FOR K=1 TO 10
 40 PRINT CSR0;"■■■■■■■■■■";
 50 A=INT(RAN#*10)
 60 PRINT CSRA;"?";
 70 FOR I=1 TO 50
 80 B$=KEY$:IF B$≠"" THEN 100
 90 NEXT I
100 IF B$<"0" THEN 130
110 IF B$>"9" THEN 130
120 IF A=VAL(B$) THEN N=N+1:BEEP 1
130 IF KEY$≠"" THEN 130
140 NEXT K
150 PRINT
160 PRINT "SCORE:";N;
170 IF N<10 THEN END
180 FOR I=1 TO 10
190 BEEP 0:BEEP 1
200 NEXT I
210 END
```

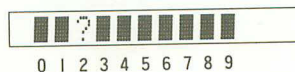
(226ステップ)

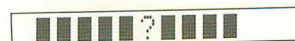
操 作 例

まずはゲームスタート。

RUN EXE

■■?■■■■■■■

0 1 2 3 4 5 6 7 8 9

"?"モグラの顔を出している位置がいくつかで、数字キーを押します。

②

今度は5の位置です。

5	■■■?■■■■■■
∫	∫
7	SCORE: 7

10匹が出終るとゲーム終了となり、得点を表示します。
効果音として、モグラをたたくと「ピッ」と音が鳴り、10匹全部たたくと連続音が鳴ります。

ここで説明する命令および関数

FOR ~ TO ~ STEP, NEXT, BEEP, KEY$, CSR, VAL, STR$

■FOR~TO~STEP, NEXT(作業を繰り返す)

FOR文はNEXT文と対になって使われ、FOR・NEXT文とも呼ばれます。
FOR文はある作業を繰り返す場合、繰り返しの回数がわかっているときに便利な命令です。

```
 30 FOR K=1 TO 10
   :
140 NEXT K
```

この使い方は、同じモグラを出させてたたかせる操作を10回繰り返すため、変数Kの内容が1から1つづつ増えていき、10をこえたときに行番号140の「NEXT K」以降に進むことを示しています。
FOR文は、

FOR 制御変数=初期値 TO 終値 STEP 刻み幅

という型式で使われ、制御変数とは行番号30の変数Kのことで、FOR文とNEXT文の間を繰り返す回数を制御する変数です。この制御変数に使えるのはA~Zまでの数値変数で、配列変数(A( )、B( )という形の変数)は使えません。
初期値、終値、刻み幅というのは何回繰り返すのか、またいくつ刻みで繰り返すのかを決めます。行番号30のように刻み幅が省略されたときは、刻み幅を1と見なします。

たとえば、10回繰り返すのであれば、

```
FOR A=1 TO 10
FOR B=0 TO 9 STEP 1
FOR C=4.5 TO 0 STEP -0.5
```

としても同じことです。
初期値は1でなくても0でもかまいませんし、刻み幅は小数でも負数でもかまいません。刻み幅が負数のときは、初期値から刻み幅を引いてゆき、終値より小さくなったら終了します。
NEXT文は、

```
NEXT 制御変数
```

という型式で使われ、FOR文で使われた制御変数と同じものを書きます。
次のプログラム例をためしてみてください。

```
例1)  10 FOR A=1 TO 11
      20 PRINT CSRA;"*";
      30 BEEP
      40 NEXT A
例2)  10 FOR A=11 TO 0 STEP -1
      20 PRINT CSRA;"*";
      30 BEEP
      40 NEXT A
例3)  10 FOR A=0 TO 11 STEP 0.5
      20 PRINT CSRA;"*";
      30 BEEP
      40 NEXT A
```

この3つの例で異なる所は行番号10のFOR文です。
例1では"*"が左端から順に表示されていき、例2では"*"が右端から順に表示されていきます。例3では左端から順に表示されていきますが、ビープ音が2回鳴るごとに表示の"*"が1つづつ増えていきます。
FOR・NEXT文で良く使われるのが、行番号180～200までのように、同じ作業を繰り返すためと、一定時間表示をそのままにしておくために使われることがあります。

```
例4)  10 PRINT "PB-110";
      20 FOR A=1 TO 100:NEXT A
      30 PRINT
      40 PRINT "END"
```

この例の行番号20が時間待ちの役目をしています。一見何の意味もなさそうですが、行番号10のPRINT文で「PB-110」を表示し、変数Aの値が1から100に1ずつ増えていく間、そのままの表示を続けます。実際にためしてみると、役目が良くわかると思います。
なお、このような使い方で注意しなければならないのが、行番号10の最後の「;」と、行番号30の「PRINT文のみ」です。行番号10のように表示後「;」をつけなければ、「PB-110」を表示後停止してしまいますので、行番号20に進まないのです。また、行番号30のように「PRINT」を入れておかなければ、行番号40の「END」が続けて表示され、「PB-110 END」となってしまうのです。
以上のように、FOR・NEXT文は同じ作業を一定回数繰り返すのに便利な命令です。

■BEEP(効果音を楽しむ)

ここまでたびたび出てきましたBEEPについて説明します。BEEP文はゲームの効果音や入力の確認などに便利な命令で、高い音と低い音の2種類が選べます。

BEEP 0……低い音
BEEP 1……高い音
※0または1を省略して「BEEP」とだけしたときは「BEEP 0」と同じです。

```
120 IF A=VAL(B$) THEN N=N+1:BEEP 1
```

行番号120のように、入力した答えが正しければ高音で知らせるとした方が、ゲームとしても楽しくなります。
また、行番号180～200のようにFOR・NEXT文と組み合わせて何回か鳴らせば、一段と効果が出ます。

■KEY$(1文字の入力)

KEY$はキーボードで押された1文字(1キー)だけを読み込む関数です。

```
20 IF KEY$≠"" THEN 20
```

この使い方は、IF文と組み合わせてキーが押されていれば行番号20を繰り返し、キーが押されていなければ次に進みます。
つまり、このプログラム実行のために「SHIFT P0」のP0キーを押し続けても先に進まないようにしているもので、KEY$関数によりキーを押している内容を読み込み、ヌル(何もないことで、キーを押していない状態。" "で表わされます)であれば次の行に進みます。なお、EXEキーを押してもヌルと見なされますので、「RUN EXE」の操作では、すぐにスタートします。行番号130も同じ使い方で、キーがはなされなければ次の問題に進まないようにするためです。

```
 80 B$=KEY$:IF B$≠"" THEN 100
  :
100 IF B$<"0" THEN 130
110 IF B$>"9" THEN 130
  :
```

行番号80～110の使い方は、まず行番号80でキーが押されたかを判断し、キーが押されていれば行番号100にジャンプします。行番号100と110は押されたキーのチェックで、⓪～⑨のキー以外を押した場合は行番号130へジャンプします。ここでの使い方が行番号20、130と異なる点は、押されたキーの内容を覚えておくかどうかです。行番号20、130ではKEY$を実行した時点のキー状態を見るだけですが、行番号80からの使い方は押されたキーの内容を次の作業で使うため、内容を覚えておかなくてはならないのです。そのために文字変数B$に代入しておきます。

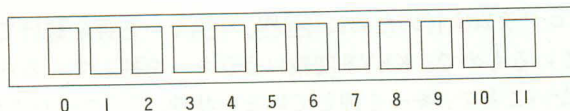
例)
```
10 A$=KEY$
20 IF A$="0" THEN BEEP 0
30 IF A$="1" THEN BEEP 1
40 GOTO 10
```

この例は、⓪キーを押せば低い音が鳴り、①キーを押せば高い音が鳴ります。このように、KEY$関数はプログラムの流れを停止させることなく、キーを押している状態を知ることができますので、リアルタイムゲーム(瞬時にキーを押して遊ぶゲーム)にはとても便利です。

■CSR(表示位置を指定する)

CSR関数はPRINT文中で使われ、次の出力要素をどこから表示するかを指定します。

```
40 PRINT CSR0;"████████████";
```

行番号40のように必ず左端から表示させたいときには「0」を指定します。この指定は表示窓の左端から0、1、2……11と数えます。

```
60 PRINT CSR A;"?";
```

行番号60のように「?」のマークを表示させる位置がそのつど変わるような場合は数値変数を使います。このときの変数の値は0から11まで(0≦変数<12)でなければなりません。

例）
```
10 X=5
20 PRINT CSRX;"*";
30 $=KEY$
40 IF $="4" THEN X=X-1:BEEP 0
   :IF X<0 THEN X=0
50 IF $="6" THEN X=X+1:BEEP 1
   :IF X>11 THEN X=11
60 PRINT
70 GOTO 20
```

このプログラムは、最初は真中に「*」が表示され、④キーを押すと「*」が左に動き、⑥キーを押すと「*」が右に動きます。

このように表示位置を左端(0)から右端(11)まで自由に指定できますが、この例のように使う場合は、行番号40、50のようにIF文による範囲の判断を必ず行なってください。行番号40では④キーが押されると表示位置を1減らして左に動かしますが、位置指定が0より小さくなりますとエラー(ERR5)になりますので、Xから1を引いた後IF文で0より小さいかどうかを判断させます。行番号50も同様に、⑥キーが押された場合はXの値に1を加え、11より大きくならないかを判断します。

もし、行番号40と50を次のように変更するとどうなるでしょうか。

```
40 IF $="4" THEN X=X-1:BEEP 0
   :IF X<0 THEN X=11
50 IF $="6" THEN X=X+1:BEEP 1
   :IF X>11 THEN X=0
```

このように変更すると、④キーを押しつづければ左端にいっても止まらずに、右端から出てきます。⑥キーを押しても同様に右端にいって、左端から出てきます。

このように、CSR関数を使えば表示位置を思う通りに指定することができます。

■VAL（数字を数値に変換する）

VAL関数とは文字変数に入っている数字(123等でABCなどの文字ではない)を数値に変換します。

```
120 IF A=VAL(B$) THEN N=N+1:BEEP 1
```

行番号120では、KEY$関数により文字変数B$に代入された数字を数値に変換し、数値変数Aの内容と比較しているのです。これはKEY$関数が1キー分の入力を文字として読み込むため、文字変数にしか代入できません。この例のように数値(変数)と比較したい場合には文字変数のままでは型式が異なるため、文字変数の内容を数値に変換してやらなければなりません。

ここで、数字と数値の違いを見てみましょう。

例）
```
10 A$="123":B$="456"
20 PRINT A$+B$
30 PRINT VAL(A$)+VAL(B$)
```

行番号10では文字変数A$とB$に文字(数字)である「123」と「456」を代入しています。行番号20ではこの2つの文字変数の内容を加えた結果を表示します。文字を加えるというのは文字を連結するという意味ですので、結果は「123456」となります。ところが行番号30では文字変数の内容を数値に変換してから加えますので、たし算ということになり、123＋456の計算結果「579」が表示されます。

"123"+"456"→"123456"
　　連結

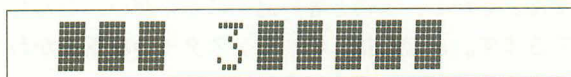
123 + 456　→ 579
　　たし算

VAL関数は文字変数内の数字を数値に変換する働きを持っています。
なお、VAL関数と対になるもので、STR$関数があります。STR$関数はVAL関数とは逆に数値を数字に変換する関数です。

■STR$(数値を数字に変換する)

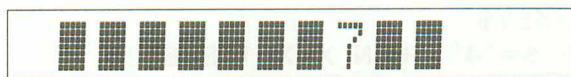
行番号60を次のように変えてみるとどうなるでしょうか。

```
60 PRINT CSRA;A;
```

これは今まで「?」が表示されていたところに答を表示させようとしたのですが、結果は……

のように表示されるはずです。これは表示される変数Aの内容が数値ですので、「3」の前に符号桁(+または-)がついて表示されてしまうのです。これでは表示もきれいではなく、位置もズレています。(「3」の表示されている位置は4なのです)

そこで次のようにしてみましょう。

```
60 PRINT CSRA;STR$(A);
```

どうなりましたか。きれいに答が入っているはずです。

このようにすれば100発100中。でも、つまらないという方は行番号70を次のように変更してください。

```
70 FOR I=1 TO 30
```

これで、違ったゲームとして楽しめます。今度のゲームは表示された数字に対応する0~9のキーを押しますが、かなり反射神経を必要とします。まだまだ簡単だと言う方は、行番号70の「30」をどんどん小さくしていってください。どんどんむずかしくなります。

4−3 宝物を取り返せ

このプログラムは、モンスターに奪い取られた宝物を、モンスターの攻撃をかわしながら取り返すゲームです。
モンスターはあなたに向っておそってきますが、最初のうちはあまりスピードが早くありません。しかし、宝物を奪い返すにしたがい、だんだん早いスピードで追いかけてきます。あなたはこの、モンスターの攻撃をかわしながら、できるだけ多くの宝物を取り返してください。

プログラムリスト

```
 10 PRINT "HI SCORE";S;
 20 GOSUB 500
 30 B=5:C=0:E=0.1:T=0:X=0:Z=0
 40 PRINT
 50 PRINT CSRX;"Ω";CSRB;"█";
 60 IF INT(B)=X THEN GOSUB 600
 70 IF C=0 THEN A=INT(RAN#*12):C=1
 80 IF C=1 THEN PRINT CSRA;"ฏ";
 90 $=KEY$
100 IF $="4" THEN X=X-1:BEEP 0
    :IF X<0 THEN X=11
110 IF $="6" THEN X=X+1:BEEP 0
    :IF X>11 THEN X=0
120 IF $≠"5" THEN 140
130 IF A=X THEN C=2:BEEP 1:C=0
    :T=T+10:E=E+0.03
140 IF X<B THEN B=B-E
150 IF X>B THEN B=B+E
160 GOTO 40
500 REM WAIT
510 FOR V=1 TO 100:NEXT V
520 BEEP 1:PRINT
530 RETURN
600 REM MONSTER
610 FOR I=1 TO 10
620 PRINT CSRX;"*";
630 BEEP 0:BEEP 1
640 PRINT CSRX;"█";
650 NEXT I
```

```
660 PRINT:PRINT "SCORE:";T;
670 GOSUB 500
680 X=0:C=0:B=5
690 IF Z<2 THEN Z=Z+1:RETURN
700 IF S>T THEN END
710 S=T:$="HI SCORE"
720 FOR I=1 TO 8
730 BEEP 1:BEEP 0:PRINT MID$(I,1);
740 NEXT I
750 PRINT S
760 END
```

※このプログラムは479ステップを必要としますので、残りステップ数を確認し、もし足りないときには他のプログラムやデータバンクデータをクリアーしてからお使いください。

操　作　例

このゲームを始める前に、最初だけメモリーをクリアーします。これは、ハイスコアーが残るようにプログラム上でCLEAR文を使っていないので、別のプログラム使用後に始めるときはCLEAR[EXE]が必要です。

CLEAR[EXE]
RUN[EXE]

まず、ハイスコアーが表示され、すぐにスタートとなります。

モンスターがあなたに向っておそってきますので、左に逃げるときは[4]キーを、右に逃げるときは[6]キーを押し続けます。

[4]

左端にいき当ったときは自動的に右端に回り込みます。

⋮

[4]

宝物の所に着いたときは[5]キーを押して宝物を取ります。宝物は1個10点です。

[5]

宝物を取ると次の宝物が現われますので、また取りに行きます。

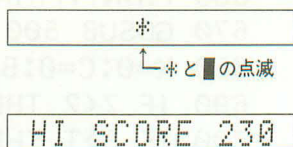
[6]

⋮

モンスターに3回おそわれるとゲーム終了になります。

```
*
```
└*と█の点滅

ゲーム終了後、あなたの得点が表示され、最高得点であれば、ハイスコアー表示が出ます。

```
HI SCORE 230
```

ここで説明する命令および関数

GOSUB, RETURN, REM, MID$

■GOSUB, RETURN(同じ作業はまとめて1つに)

GOSUB文はRETURN文とペアで使われ、同じ作業が何回も、色々な箇所で使われる場合にその作業をまとめて、効率よくプログラムを作るために使われます。

```
 20 GOSUB 500
  :
670 GOSUB 500
```

このように、行番号500～530は2箇所で使われる作業で、本当は行番号20と行番号670に同じものが入っていたものを独立させて両方で使えるようにしています。このように独立させて共通にしたものを**サブルーチン**といいます。これは、今迄でてきたプログラムのように全体の流れとともに作業をするプログラムをメインルーチンと呼ぶのに対してこう呼ばれます。

行番号20を実行すると、一度行番号500に分岐し、行番号510、520を実行後、行番号530のRETURN文により分岐してきた所の次の文に戻ります。

GOSUB文はサブルーチンへ分岐せよという命令で、RETURN文は分岐してきた所の次の文へ戻れという命令です。

このプログラムの行番号500からのサブルーチンは、表示をある一定時間止めておくためにFOR・NEXT文を使って作られています。

つまり、変数Vの値が1から100になるまでFOR・NEXT文を繰り返すので、時間かせぎができるのです。

また、行番号600からのサブルーチンは、前者と少し違っています。それは前者のサブルーチンは必ず元のGOSUB文の次に戻るのに対し、このサブルーチンは戻らずにプログラムを終了することがあります。

GOSUB文の次に戻るためにはRETURN文が必要ですが、RETURN文の入っている行を見るとIF文になっています。このサブルーチンではIF文の条件を満たし、モンスターにおそわれた回数が2回以内ですとRETURN文により戻りますが、3回になると戻らずにプログラム終了の方向へ進みます。

GOSUB文による分岐は必ずRETURN文により戻すのが原則ですが、このように、そのままプログラムを終了することもあります。

サブルーチンを使うときに1つ気を付けなければならないのが、メインルーチンとの境目です。この例では行番号160がGOTO文になっていますので、必ず先頭の方へ戻りますが、もしこのようなGOTO文がなく、END文もない場合はどうなるでしょうか。

例)

```
 10 PRINT "LINE 10"
 20 GOSUB 100
 30 PRINT "LINE 30"
 40 GOSUB 100
100 PRINT "LINE 100"
110 RETURN
```

この例では行番号100と110がサブルーチンですが、境目にEND文が入っていません。このまま実行すると次のようになります。

操作	表示
RUN EXE	LINE 10
EXE	LINE 100
EXE	LINE 30
EXE	LINE 100
EXE	LINE 100
EXE	ERR7 P0-110

これは、行番号40を実行後プログラムを終了せずに、行番号100からのサブルーチンへ進み、行番号110のRETURN文により戻り先がわからないためにエラーとなったのです。

このような場合は、行番号50としてEND文を入れると正しく終了します。

操作	表示
AC MODE 1	P _123456789
50 END EXE	50 END
MODE 0 RUN EXE	LINE 10
EXE	LINE 100
EXE	LINE 30
EXE	LINE 100
EXE	_

サブルーチンの使い方として、共通部分を抜き出して繰り返して使うことがありますが、この他に、ある1つの作業をするブロックとしてサブルーチンを作り、あとでつなげて1つのプログラムにする使い方があります。
この使い方の例は第6章にまとめて「便利なサブルーチン集」としてありますので、くわしくは第6章をご覧ください。

■REM(注釈をつける)

REM文は実行しても何もしない命令ですが、見出しや注釈をつけてプログラムを見やすく、わかりやすくする役目を持っています。

```
500 REM WAIT
  :
600 REM MONSTER
```

このようにしておけば、行番号500からは時間待ちのサブルーチンであるとか、行番号600からはモンスターにおそわれたときの処理をするサブルーチンであるということが、一目でわかります。
また、プログラムの先頭につけてこのプログラムが何のプログラムであるかを区別するときにも使われます。

例) 5 REM *MONSTER GAME*

REM文により後に書かれている内容は、全て注釈として扱われますので、プログラム実行に必要な代入文や命令は続けて書かないようにしてください。

例) 5 REM *GAME*:A=0

└実行されない。(:A=0 の部分)

■MID$(文字列を取り出す)

MID$関数は、専用文字変数($)に記憶されている文字列の中からいくつかを取り出す文字関数です。

```
710 S=T:$="HI SCORE"
720 FOR I=1 TO 8
730 BEEP 1:BEEP 0:PRINT MID$(I,1);
740 NEXT I
```

この例では専用文字変数に「HI SCORE」という文字列を代入し、FOR・NEXT文により順番に1文字づつ取り出して表示させます。

MID$(位置,文字数)

- 文字数：取り出す文字数
- 位置：文字を取り出す位置(左から数える)

MID$関数では、左から何文字目の位置から何文字取り出すかを指定します。たとえば、専用文字変数に「MONSTER」を代入し、いくつかを取り出してみます。

例）
```
10 $="MONSTER"
20 PRINT MID$(1,3)……左から1文字目より3文字取り出す。
30 PRINT MID$(2,4)……左から2文字目より4文字取り出す。
40 PRINT MID$(3,7)……左から3文字目より7文字取り出す。
50 PRINT MID$(8,2)……左から8文字目より2文字取り出す。
60 PRINT MID$(1)  ……左から1文字目以降全部を取り出す。
```

このプログラムを実行すると次のようになります。

操　作	表　示
RUN EXE	MON
EXE	ONST
EXE	NSTER
EXE	
EXE	MONSTER

最初の行番号20では左から1文字目より3文字取り出しますので、「MON」となります。次の行番号30では左から2文字目より4文字取り出しますので「ONST」となります。行番号40では左から3文字目より7文字取り出しますが、3文字目からは5文字しかありませんので、残り全部を取り出します。行番号50では左から8文字目より2文字取り出しますが、全部で7文字しかありませんので、取り出す文字は無しとなります。行番号60では今迄のとは少し異なり何文字取り出すかの指定がありません。このような場合には指定位置、ここでは左から1文字目より最後までを取り出します。
このように、MID$関数は専用文字変数($)に記憶されている文字列の中から何文字かを取り出す働きを持っています。

4-4 あっち向いてホイ！ゲーム

このプログラムは反射神経を競うゲームです。
上下左右を向く矢印に合わせて②、④、⑥、⑧のキーを素早く押します。
押し間違えたり押さなかったときはアウトとなり、3回アウトになるとゲームオーバーです。
なお、だんだん早く押さなければなりませんので、ガンバッテください。

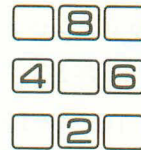
プログラムリスト

```
 10 CLEAR
 20 DEFM 4
 30 FOR A=1 TO 4
 40 READ Z$(A)
 50 NEXT A
 60 DATA ↓,←,→,↑
 70 F=20
 80 PRINT:BEEP 1
 90 PRINT CSR1;"<<   >>";
100 FOR A=1 TO 5            3文字分のスペース
110 M=INT(RAN#*4+1)
120 PRINT CSR4;"■";:BEEP 0
130 FOR B=1 TO F:NEXT B
140 PRINT CSR4;"□";:BEEP 1
150 NEXT A
160 PRINT CSR4;Z$(M);
170 FOR A=1 TO F*2
180 $=KEY$:IF $≠"" THEN 210
190 NEXT A
200 GOTO 500
210 IF M≠VAL($)/2 THEN 500
220 G=G+10:F=F-1
230 GOTO 80
500 REM *OUT*
510 FOR B=1 TO 10
520 PRINT CSR4;"*";CSR9;"OUT";
530 BEEP 1
540 PRINT CSR4;"+";
550 NEXT B
```

```
560 H=H+1:IF H<3 THEN 80
570 PRINT
580 PRINT "SCORE";G;
590 BEEP 0:BEEP 1
600 END
```

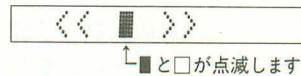
プログラム 338ステップ
増設メモリー 32ステップ
計 370ステップ

操　作　例

まずは表示に対するキーを覚えてください。

↑、←、→、↓の表示に合わせて素早く8、4、6、2のキーを押します。
では、ゲームスタート！

RUN EXE

<< ▮ >>

└▮と□が点滅します

点滅を繰り返し、↑、←、→、↓のいずれかが表示されます。

<< ↓ >>

素早く対応する2キーを押します。

2

<< ▮ >>

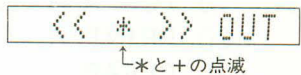
⋮

<< + >>

だんだん早く押さなければならなくなります。遅れたり間違ったキーを押すとアウトになります。

8

<< * >> OUT

└＊と＋の点滅

アウトを3回するとゲームオーバーとなりスコアーが表示されます。

SCORE 120

ここで説明する命令および関数

CLEAR, DEFM, READ, DATA, RESTORE

■CLEAR(変数内容を消去する)

CLEAR文は変数の内容をクリアーにする命令で、A～Zの変数および、増設した変数全てに0を代入します。専用文字変数($)は何も入っていない状態(ヌルという)になります。

```
10 CLEAR
```

このプログラムのようにハイスコアーを残さないのであれば、全部の変数を一度にクリアーできた方が便利です。そのためにプログラムの先頭にCLEAR文を入れておきます。

例)
```
10 A=123:B$="ABC"
20 PRINT A;B$
30 CLEAR
40 PRINT A;B$
```

この例では数値変数Aに123を、文字変数B$に「ABC」を代入しておき、CLEAR文を実行する前と後の変化を見ます。
このプログラムを実行しますと、

操作	表示
RUN EXE	123ABC
EXE	0

このように一度記憶させた内容をクリアーしてしまいますので、プログラムの途中で使うときには注意してください。

〈注意〉
CLEAR文は変数全てをクリアーしてしまうため、FOR・NEXT文で使うと制御変数もクリアーしてしまいますので、FOR・NEXT文中では使えません。

例)
```
10 FOR A=1 TO 10
20 PRINT A
30 CLEAR
40 NEXT A
```

この例でためしてください。1回目で「1」を表示後「ERR7」となります。これはCLEAR文により制御変数の内容がクリアーされ、FOR・NEXT文中であることがわすれられてしまうのです。
CLEAR文を使う場合はなるべくプログラムの先頭の方に入れるようにしてください。

■DEFM(メモリーを増やす)

変数には最初のA～Zの26個と、専用文字変数($)がありますが、これより多くの変数が必要となることもあります。このように多くの変数を必要とするときにDEFM文を使います。

```
20 DEFM 4
```

DEFM文に続く数値は増やす個数で、この場合は4個増やしますので、最初の26個と合わせて30個となります。

DEFM文はプログラムとして書き込んでも使えますが、マニュアルでも使うことができます。

例）10個増やして、合計36個にする。

```
DEFM 10 EXE          ***VAR:36
```

また、現在の変数の数がいくつになっているかを確認したい場合は、個数を省略します。

```
DEFM EXE             ***VAR:36
```

変数の数を変えたい場合は再びDEFM文を実行しますが、26個より減らすことはできません。最初の26個に戻したいときは0個を指定します。

```
DEFM 0 EXE           ***VAR:26
```

DEFM文の使い方としては大きく分けて3つあります。

1つ目は増設する変数の数を指定する。2つ目は変数の数を確認する。3つ目は変数の数を最初の26個に戻す。となります。

DEFM文をマニュアルで実行させても、プログラム中に書き込んで実行させても、働きは同じですが、表示が違ってきます。

次の例を見てください。

例） **〈マニュアル実行〉**

```
DEFM 5 EXE           ***VAR:31
DEFM EXE             ***VAR:31
```

〈プログラム実行〉

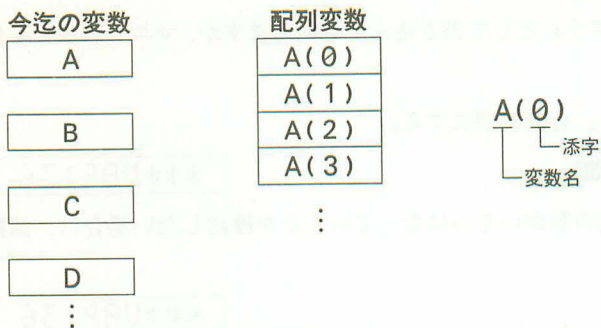
```
10 DEFM 5
20 DEFM
30 PRINT "END"
```

```
RUN EXE              表示は止まりません― ***VAR:31
                                         END
```

マニュアルで実行したときは、そのつど変数の数を表示しますが、プログラムで実行したときは、変数の確認(DEFMのみ)のときだけ表示し、停止しません。DEFM文により増えた変数は最初の26個の延長として、**配列変数**で使います。

●配列変数

ここで、配列変数について説明しましょう。

配列変数とは同じ変数名でありながら添字と呼ばれる番号をつけて使われる変数です。

このように、今迄使ってきた変数はA、B、Cのように違った名前で呼ばれて使いわけられていましたが、配列変数は名前は同じだが、使いわけるための番号がつけられています。

この番号が重要な意味を持っているのです。

実は、この番号に変数を使えるため、見かけは同じ名前でもちゃんと使いわけができるのです。

では、実際にプログラムとしたときの違いを見てみましょう。

例1） 10個の変数に1～10の値を代入する。

今迄の変数使用

```
10 A=1:B=2:C=3:D=4:E=5
20 F=6:G=7:H=8:I=9:J=10
```

配列変数使用

```
10 FOR Z=0 TO 9
20 A(Z)=Z+1
30 NEXT Z
```

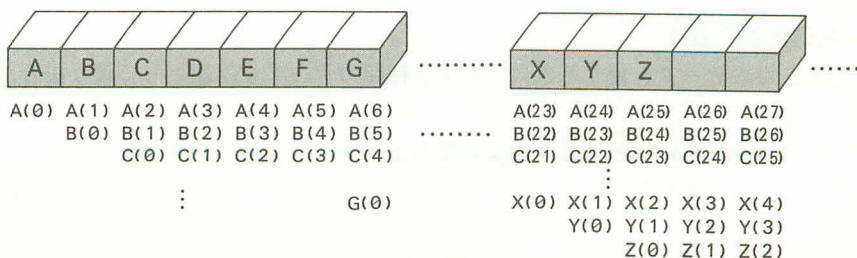
このように、いちいち「A＝」とか「B＝」のようにしていてはめんどうです。この代入の場合はこれ位ですみますが、もし何番目かの指定をして、任意の変数内容を見たいときはどうでしょう。

今迄の変数使用

```
100 INPUT Z
110 IF Z=1 THEN PRINT A
120 IF Z=2 THEN PRINT B
130 IF Z=3 THEN PRINT C
140 IF Z=4 THEN PRINT D
150 IF Z=5 THEN PRINT E
160 IF Z=6 THEN PRINT F
170 IF Z=7 THEN PRINT G
180 IF Z=8 THEN PRINT H
190 IF Z=9 THEN PRINT I
200 IF Z=10 THEN PRINT J
```

配列変数使用

```
100 INPUT Z
110 PRINT A(Z-1)
```

このような場合はかなり違います。今迄出てきた変数を使ったときは、IF文を並べて判断させなければならないのに対し、配列変数ではカッコ「(　)」内の変数の値を変えてやるだけで簡単に選ぶことができます。
配列変数は数値変数としても文字変数としても使えますが、変数名が同じであれば同じ箱を使っていますので、同時に同じ変数名は使えません。
また、変数名が違っても同じメモリーの箱を使うことがありますので、変数の重なり合いにも注意が必要です。

基本となるメモリーの箱はAからZまでの26個とDEFM文により増設されたメモリーの箱となりますので、変数名をAやB、XやZと変えても、同一の箱を使うことがあります。

〈ご注意〉

次のような使い方は配列変数と一般の変数を重複して使っていますので、このような使い方をしないように注意してください。

例)
```
10 FOR I=1 TO 5
20 F(I)=I
30 NEXT I
40 PRINT F(1);F(2);F(3);F(4);F(5)
50 END
```

このプログラムは配列変数F(1)～F(5)に1、2、3……5と順に数値を入れようとして使いましたが、実行すると次のようになります。

RUN EXE　　　　1 2 6 4 5

3番目のデータが入っているF(3)の内容が正しくありませんね。
正しくは「3」が入っていなければなりませんのに、「6」が入っています。
これは、変数の使い方が次のようになっているからです。

A	B	C	D	E	F	G	H	I	J	K	L……
						F(1)	F(2)	F(3)	F(4)	F(5)	………

このように、配列変数F(3)は変数Iと同じ箱を使っています。この例では配列変数F(1)～F(5)のほかに変数IをFOR・NEXTループの制御変数として使っています。変数Iの内容は1から順に2、3、4、5と変化していき、6になると終値の5をこえたと判断してFOR・NEXTループを終了し、次の行番号40へ進みます。
このように、変数Iの内容が6ですので、配列変数F(3)の内容も6となってしまうのです。
配列変数を使うときにはこのような変数の重なり合いをさけて使わなければなりません。配列変数と一般の変数(A～Z)の関係は巻末(168ページ)に一覧表がありますので、参照してください。

〈配列変数の応用〉

たくさんのデータを使うときに、必ず2種類のデータを1組として使う場合があります。
このようなときにはどうしたらよいでしょうか?
次の例をみてください。

例）15人の名前と身長を記憶させる。

```
P0   10 CLEAR:DEFM 6
     20 FOR A=1 TO 15
     30 INPUT "NAME",B$(A)
     40 INPUT "SHINCHO",B(A+15)
     50 NEXT A
     60 END

P1   10 INPUT "NAME",B$
     20 FOR A=1 TO 15
     30 IF B$=B$(A) THEN PRINT
        B(A+15);"cm":GOTO 20
     40 NEXT A
     50 PRINT "NO NAME"
     60 GOTO 10
```

P0のプログラムは入力プログラムで、名前と身長を入力します。
P1のプログラムは名前を入力して、該当する人の身長を表示させるプログラムです。
ここでの変数の使い方は、FOR・NEXTループの制御用として変数Aを使い、名前をさがすときの一時記憶用に変数B$を使っています。その他の変数は全て名前と身長を記憶させる配列として使います。

A	B	C	D	E	F	G	H
ループ用	B$	B$(1)	B$(2)	B$(3)	B$(4)	B$(5)	B$(6)
I	J	K	L	M	N	O	P
B$(7)	B$(8)	B$(9)	B$(10)	B$(11)	B$(12)	B$(13)	B$(14)
Q	R	S	T	U	V	W	X
B$(15)	B(16)	B(17)	B(18)	B(19)	B(20)	B(21)	B(22)
Y	Z	Z(1)	Z(2)	Z(3)	Z(4)	Z(5)	Z(6)
B(23)	B(24)	B(25)	B(26)	B(27)	B(28)	B(29)	B(30)

B$(1)からB$(15)(C～Q)までは文字配列変数として使い、名前を記憶させます。B(16)からB(30)(R～Z(6))までは数値配列変数として使い、身長を記憶させます。
このように2組のデータを同時に処理するような場合には、1つめの配列変数は1から15まで使い、もう1つの配列変数は同じ変数名で15個あとの16から30までを使うことにより行なわれます。
なお、P0の行番号10のDEFM文は、標準状態の変数がA～Zまでの26個ですので、この例で使われる32個に合わせるために変数を6個増やします。

■READ・DATA・RESTORE(データを読み込む)

今迄データを変数に代入するには、代入文を使ってきました。しかし、データの数が多くなってきますと、いちいちめんどうなものです。
そこで、プログラム中にデータを書き込み、自動的にデータを変数に代入するのが、READ・DATA文であり、どのデータを読み込むかの位置を指定するのがRESTORE文です。

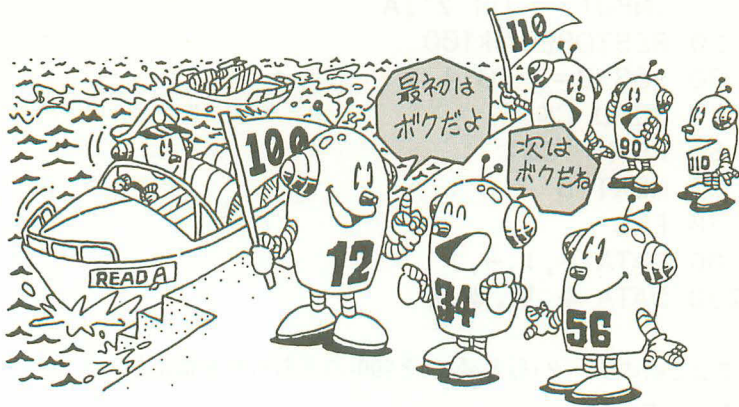
```
30 FOR A=1 TO 4
40 READ Z$(A)
50 NEXT A
60 DATA ↓,←,→,↑
```

READ文はDATA文にあるデータをもってきて、READの後に続く変数に入れます。
この例ではFOR・NEXT文により4回繰り返しREAD文を実行します。データをもってくる順番は1回目が「↓」、2回目が「←」、3回目が「→」、4回目が「↑」となります。
ここでは文字データを文字変数に入れていますが、数値データを数値変数に入れるときも同様です。

例)
```
10 FOR A=1 TO 5
20 READ B
30 PRINT B
40 NEXT A
50 END
60 DATA 12,34,56,78,90,
```

この例では、データは先頭から順に読み込まれ、表示されます。

操作	表示
RUN EXE	12
EXE	34
EXE	56
EXE	78
EXE	90

データが読み込まれる順番は、プログラム実行時に一番先頭のデータからとなりますので、DATA文がいくつもあるときは行番号の小さい方から、また一行のDATA文中では先頭(左端)からとなります。
DATA文からデータが読み込まれますと、次のREAD文で読み込まれるデータは、今読み込んだデータの次のデータとなります。

READ文では変数を，(カンマ)で区切ることにより、同時にいくつも読み込むことができます。

例）
```
10 FOR A=1 TO 4
20 READ B,C
30 PRINT B+C
40 NEXT A
50 END
60 DATA 14,25,36,47,58,69,42,53
```

この例では同時に2つの変数に読み込んでいます。このように必ずいっしょに使うデータであれば、，で区切って読み込んだ方が便利です。

操　作	表　示
RUN EXE	39
EXE	83
EXE	127
EXE	95

今迄の例ではプログラムを実行開始した後、先頭のDATA文から順にデータを読み込みましたが、もし同じデータを2回以上使いたかったり、条件に応じてデータを変えたい場合はどうでしょう。
実は、このデータの位置を指定する命令がRESTORE文なのです。

まずは次の例を見てください。

例）
```
 10 INPUT "1 OR 2",A
 20 RESTORE A*100
 30 FOR B=1 TO 4
 40 READ C$
 50 PRINT C$;
 60 NEXT B
 70 END
100 DATA ←,↓,→,↑
200 DATA ♠,♥,♦,♣
```

この例では2組のデータ（行番号100と200）のどちらかを指定し、4つのデータを表示させます。
行番号10でデータの選択をし、1か2の数値を入力します。次の行番号20のRESTORE文でA＊100すなわち、Aが1のときは100を、Aが2のときは200を指定することになり、行番号100と200の選択を行ないます。

操　作	表　示
RUN EXE	1 OR 2?
2 EXE	♠♥♦♣

このように、RESTORE文は次に続く数式の値により次に読み込むDATA文の行番号を指定します。
行番号の指定には直接行番号の数値を書き込んで使ったり、代入された数値変数や計算式を使うことができますが、必ずDATA文のある行番号を指定してください。
RESTORE文による指定行が存在しないときはエラー（ERR4）となります。指定行はあるがDATA文ではないときはエラーとはならず、指定行以降にある一番近いDATA文を指定します。
なお、RESTORE文の行指定は省略でき、省略した場合はそのプログラムエリアの最初に出てくるDATA文を指定します。

4−5 その他の命令および関数

これまで、例題のプログラムとともにいろいろな命令や関数を説明してきましたが、ここでは今迄に出てこなかった命令や関数について説明します。

■MODE（計算機の状態を設定する）

MODE文は次に続く値により、次の状態を設定します。

MODE　4……角度単位を「度」(DEG)に設定します。
MODE　5……角度単位を「ラジアン」(RAD)に設定します。
MODE　6……角度単位を「グレード」(GRA)に設定します。
MODE　7……プリントモードに設定します。
MODE　8……プリントモードを解除します。

MODE 4 ～ 6の角度単位は三角関数計算(24ページ)をするときに必要となります。

MODE 7はプリントモードの設定で、以後、操作内容や演算結果の表示、プログラムリストなどがプリンタに印字されます。このプリントモードはMODE 8で解除されます。

例）
```
10 INPUT A
20 B=A*A
30 MODE 7
40 PRINT B
50 MODE 8
60 END
```

MODE 7またはMODE 8は、このようにプログラム中に書き込んで使えば、演算結果だけを印字させることができます。

このMODE文はプログラム中に書き込む場合、MODEキーではなく、アルファベットキーのMODEを使って書き込みます。

MODE 7、MODE 8と押してマニュアルでプリントモードを設定することもできます。プログラムリストやマニュアル計算の内容を印字させるときにはMODE 7、MODE 8と押し、プログラム中で答えだけを印字させるときには「MODE 7」、「MODE 8」と書き込みます。

■SET（小数点以下指定、有効桁数指定）

SET文は表示内容を小数点以下を指定したり、有効桁数を指定したりする命令です。変数内には全桁入っています。

SET Fn……小数点以下指定（$0 \leqq n \leqq 9$。n は整数）
SET En……有効桁数指定　（$0 \leqq n \leqq 9$。n は整数）
SET N　……指定解除

例）　1234.56789
　有効桁数3桁↑　小数点以下3桁↑

SET文により指定しますと、次のPRINT文による表示やマニュアル計算の答えの表示は指定に従っています。
SET Fnによる指定は、小数点以下n桁までを四捨五入で求めるよう指定します。

例）
```
10 A=12.3456789
20 SET F0
30 PRINT A
40 SET F2
50 PRINT A
60 SET F5
70 PRINT A
80 SET N
90 PRINT A
```

この例は「12.3456789」という数値を小数点以下0桁、2桁、5桁までの表示と、全ての表示を行なっています。

操　作		表　示
RUN EXE	（小数点以下0桁）	12
EXE	（小数点以下2桁）	12.35
EXE	（小数点以下5桁）	12.34568
EXE	（全桁表示）	12.3456789

SET Enによる指定は、有効桁数n桁までを四捨五入で求めるよう指定します。
有効桁数とは左端から数えて何桁かということです。
なお、有効桁数0を指定しますと、10桁が指定されます。

例)
```
10 A=123.456789
20 SET E2
30 PRINT A
40 SET E5
50 PRINT A
60 SET E0
70 PRINT A
80 SET N
90 PRINT A
```

この例は「123.456789」という数値を有効桁数2桁、5桁、10桁までの表示と、全ての表示を行なっています。

操　作		表　示	
RUN EXE	(有効桁数2桁)	1.2E02	(1.2×10^2)
EXE	(有効桁数5桁)	1.2346E02	(1.2346×10^2)
EXE	(有効桁数10桁)	1.234567890E02	(1.234567890×10^2)
EXE	(全桁表示)	123.456789	

SET文により桁数指定を行なった後は、必ず「SET N」により指定を解除してください。指定を解除しない場合は以後のPRINT文による表示やマニュアル計算による答えの表示にも指定が有効となります。

■LEN(文字変数内の文字数を数える)

LEN関数は文字変数内に記憶されている文字列の長さを数える関数です。

例)
```
10 INPUT A$
20 PRINT LEN(A$)
30 GOTO 10
```

行番号10で入力されたA$内の文字列の長さを表示します。文字変数はA$とかB$のような文字変数や専用文字変数($)が使えます。
このLEN関数を使えばいろいろ便利なことができます。

例) 3桁位取りをつける。
```
10 INPUT $
20 A=LEN($)
30 B=INT(A/3)
40 C=A-B*3
50 IF A≦3 THEN PRINT $:GOTO 10
60 IF C=0 THEN 90
70 PRINT MID$(1,C); ",";
```

```
 80 IF B<2 THEN 120
 90 FOR D=0 TO B-2
100 PRINT MID$(C+1+D*3,3);",";
110 NEXT D
120 PRINT MID$(A-2,3)
130 GOTO 10
```

このプログラムは行番号10で入力された数値を文字として扱い、下桁から計算して3桁ごとに位取りの「,」をつけて表示します。
行番号20では入力された文字列の長さ(文字数)を求めます。行番号30と40で3桁区切りの数と余りの文字数を求めます。行番号50では文字数が3桁以内であれば「,」を付ける必要がありませんので、そのまま表示して入力に戻ります。
行番号60と70では3桁で区切った場合の余り桁数があれば先に表示させ、ないときは次の行番号90からの表示に進みます。行番号80では「12345」のように「12,」と「345」で表示が終るようなときは行番号120に進み、最後の3桁を表示させます。行番号90から110のFOR・NEXT文により、繰り返して3桁の数字と「,」を表示させます。
このプログラムは実用としても使えると思いますので、サブルーチンとして他のプログラムと組み合わせても便利だと思います。

操　作	表　示
RUN EXE	?
12345 EXE	12,345
EXE	?
147852369 EXE	147,852,369
EXE	?
963 EXE	963

第5章

コマンド・リファレンス

※ここからは、文法上の繰り返し等を説明するために、以下の記法を用います。

- **太字の語**――――必ず、その通り書かなければいけない語。
- {××××／○○○○}―{ }の中の一つを選択して書かなければなりません。
- [○○○○]―[ ]の中は省略する書き方もあります。
- ○○○○*――右肩に*がついた要素は繰り返して書くことができます。
- 数式――――――10、2+3、A、S*Q等の数値、計算式、数値変数。
- 文字式―――――"ABC"、X$、N$+M$等の文字定数、文字変数、文字計算式。
- パラメータ――コマンドに伴う要素。
- Ⓟ――――――――プログラム中でのみ実行可能。
- Ⓜ――――――――マニュアルでのみ実行可能。
- Ⓐ――――――――プログラム中でもマニュアルでも実行可能。
- Ⓕ――――――――関数命令。プログラム中でもマニュアルでも実行可能。

〈例〉

DATA〔データ〕〔,〔データ〕〕*

全ての要素に〔 〕がついていますから、"DATA"とだけ書くこともできます。また、,〔データ〕に〔 〕*がついていますから、この要素は繰り返して書いてもよいことになります。従って"DATA データ, データ, ……"と書くことができます。最初の〔データ〕の部分を省略すれば、"DATA ,データ,データ,……"と書くこともできます。

GOTO {行番号／#プログラムエリア番号}

これは以下のような2通りの書き方を表わしています。

①GOTO 行番号
②GOTO #プログラムエリア番号

NEW〔ALL〕 (ニュー〔オール〕) Ⓜ

機能 プログラムの消去。プログラムと変数の消去。

パラメータ ALL指定したときはP0～P9の全プログラムと変数を消去します。

説明 (1)ALL指定がないときは、現在指定されているプログラムエリアのプログラムを消去します。変数は消去されません。

(2)ALL指定があるときは全プログラムエリアのプログラムと変数を消去します。DEFMによる設定も解除され、初期の26メモリーになります。

(3)パスワード設定中は実行できません。

(4)プログラム中に書き込んで使うことはできません。

(5)WRTモードでのみ実行できます。

※NEW ALLはNEW Aと省略することもできます。

例 MODE 1 NEW EXE

RUN〔実行開始行〕 (ラン) Ⓜ

(実行開始行＝行番号)

機能 プログラムの実行。

パラメータ 実行開始行：行番号

説明 (1)指定した実行開始行(省略した場合はプログラムの先頭)からプログラムを実行します。

(2)指定した行番号が存在しないときは、指定より大きく、かつ一番近い行から実行を開始します。

(3)変数はクリアーされません。

例

```
10 PRINT "LINE 10"
20 PRINT "LINE 20"
30 END
```

入力	表示
RUN EXE	LINE 10
RUN 20 EXE	LINE 20

LIST {行番号 / ALL} （リスト）

機　能　プログラムの内容を表示します。

パラメータ　行番号：表示する最初の行番号。

ＡＬＬ：Ｐ０からＰ９までの全プログラム内容を順に表示します。

説　明

Ｉ．RUNモードの場合

(1)行番号が指定されたときは指定行番号から、行番号が省略されたときは先頭の行から順にプログラム内容を表示します。

(2)プログラム内容は順に自動的に表示されますので、止めたいときは[STOP]キーを押します。再び次の行以降を表示させたいときは[EXE]キーを押します。

(3)プリンタ接続時のプリントモード("PRT"点灯中)では表示は止まらず順次、早い速度で表示します。

II．WRTモードの場合

(1)行番号が指定されたときは指定行番号から、行番号が省略されたときは先頭の行からプログラム内容を表示します。

(2)WRTモードでは１行ごとに表示して、編集可能状態となりますので、編集が必要ないときはそのまま[EXE]キーを押しますと次の行へ進みます。なお[SHIFT]キーに続けて押しますと、直前の行に戻ります。

●ALL指定があるときはＰ０から順にＰ９までの全プログラム内容を表示して行きます。このときはWRTモードでも順に送られ、編集することはできません。

●パスワード設定中は実行できません。

※LIST ALLはLIST Aと省略することもできます。

例

```
LIST [EXE]
LIST 30 [EXE]
```

PASS "パスワード" (パス) Ⓜ

文字列

機能　パスワードを設定または解除します。

パラメータ　パスワード：１～８文字の文字列

説明

(1)パスワードが設定されていないときにこの命令を実行しますと、全プログラムエリア(P0～P9)にパスワードが設定されます。

(2)パスワードが設定されているときにこの命令を実行しますと、現在設定されているパスワードと後から実行したパスワードが一致したときのみ、パスワードが解除されます。パスワードが一致しないときはプロテクトエラー(ERR8)となります。

(3)パスワードは１～８文字の文字列で構成され、スペース、アルファベット、数字、特殊記号などが使えます。但し、「"」自身は使えません。

(4)パスワードが設定されているとき、LIST、LIST ALL、LIST#、NEW、NEW ALL、NEW#などのコマンドは使えず、またWRTモードでの行番号EXEもエラー(ERR8)となり実行できません。

(5)プログラム中では使えません。

(6)電源スイッチオフでもパスワードは保持されます。

(7)パスワードが設定されているときに、SAVEまたはSAVE ALL文によりプログラムをテープに記録しますと、パスワードも同時に記録されます。パスワードが設定されているプログラムを、テープからLOADまたはLOAD ALL文により読み込んだ場合は、プログラムについているパスワードが設定されます。なお、現在パスワードが設定されているときに、異なるパスワードのついたプログラムをテープから読み込むことはできません。(ERR8)

注意　パスワード設定後にパスワードを忘れてしまったときは、本体裏面のオールリセットボタンを押して、全プログラムとメモリーをクリアーしてください。

例　PASS"CASIO"EXE

SAVE〔ALL〕〔"ファイル名"〕(セーブ〔オール〕) Ⓜ

文字列

機能 プログラムをカセットテープに記録します。

パラメータ ALL：全プログラムエリアのプログラムを記録。
ファイル名：1～8文字の文字列。省略可。

説明 (1)ALLが省略された場合は、現在指定されているプログラムエリアの内容を記録します。

(2)ALLがついた場合は、P0からP9までの全プログラムエリアの内容を記録します。

(3)パスワードが設定されている場合は、パスワードをつけて記録しますので、LOADコマンドにより読み込んだときに同じパスワードがつきます。

※SAVE ALLはSAVE Aと省略できます。

例

```
SAVE EXE
SAVE "CASIO" EXE
SAVE ALL "PB" EXE
```

LOAD〔ALL〕〔"ファイル名"〕Ⓜ
文字列　(ロード〔オール〕)

機　能　プログラムをカセットテープから読み込みます。

パラメータ　ALL：全プログラムエリアのプログラムを読み込む。
ファイル名：1～8文字の文字列。省略可。

説　明　(1)ALLが省略された場合は、現在指定されているプログラムエリアに"**SAVE**"で記録されたプログラムを読み込みます。

(2)ALLがついた場合は、P0からP9までのプログラムエリアに"**SAVE ALL**"で記録されたプログラムを読み込みます。

(3)パスワードつきで記録されたプログラムを読み込みますと、記録したときと同じパスワードが設定されます。

※LOAD ALLはLOAD Aと省略できます。

SAVEとLOADの関係

	LOAD	LOAD "ファイル名"	LOAD ALL	LOAD ALL "ファイル名"
SAVE	○	×	×	×
SAVE "ファイル名"	○	○	×	×
SAVE ALL	×	×	○	×
SAVE ALL "ファイル名"	×	×	○	○

但し、ファイル名は同一のものです。　○…読み込める
×…読み込めない

VERIFY 〔"ファイル名"〕 Ⓜ
文字列

(ベリファイ)

機　能　カセットテープ上のプログラムやデータの記録状態をチェックします。

パラメータ　ファイル名：1～8文字の文字列。省略可。

説　明

(1)ファイル名が指定された場合は、同一ファイル名のファイルをチェックします。

(2)ファイル名が省略された場合は、コマンド実行後に最初に現われたファイルをチェックします。

(3)チェック方式は、記録状態の型式をチェックします。(パリティーチェックといいます)

例

VERIFY EXE
VERIFY "PROG1" EXE

CLEAR Ⓐ

(クリアー)

機　能　全ての変数をクリアーします。

説　明

(1)全ての変数をクリアーし、数値変数には0を、文字変数にはヌル(何もない)を入れます。

(2)このコマンドはプログラム中に書き込んでも、マニュアルでも使えます。

(3)FOR・NEXTループ(102ページ参照)中ではループ制御変数もクリアーするために、NEXT文実行時にエラーとなります。

※CLEARコマンドはVACとしても同じに使えます。

END (エンド) Ⓟ

機　能　プログラムの実行を終了します。

説　明　プログラムの実行を終了しますので、次にプログラムがあっても実行しません。

STOP (ストップ) Ⓟ

機　能　プログラムの実行を一時停止します。

説　明

(1)プログラムの実行を一時的に停止し、"STOP"を表示して入力待ちとなります。

(2)停止はEXEキーにより実行を再開します。

(3)STOP文により停止しているときに、STOPキーを押しますと、プログラムエリアと行番号を表示します。

(4)STOPによる停止中はEXEキーによる計算が行なえます。

〔LET〕 {数値変数＝数　式 / 文字変数＝文字式} Ⓟ

(レット)

機　能　左辺の変数に右辺の式の値を代入します。

説　明　(1)数値型変数には数値式が、文字型変数には文字式が対応します。

(2)LETは省略することができます。

例

```
10 LET X=12
20 LET Y=X↑2+2*X-1
30 PRINT Y
40 A$="CASIO"
50 B$="PB-110"
60 PRINT A$;B$
70 END
```

REM 注　釈 / 文字列 Ⓟ

(リマーク)

機　能　注釈を表わす文です。

説　明　(1)プログラム中に書き込み、REM以降は注釈文として扱われ何も実行されません。

(2)同一行内に実行させたいコマンドを書き込む場合は、REM文の前にマルチステートメント(：コロン)を書いてください。

例

```
10 INPUT "R",R
20 S=π*R↑2:REM MENSEKI
30 PRINT S
40 END
```

INPUT [“メッセージ文”(文字列),] 変数名 [, [“メッセージ文”(文字列),] 変数名]* Ⓟ

(インプット)

機能　キーボードからデータを変数に入力します。

パラメータ　メッセージ：文字列。
変数名：数値変数名または文字変数名。

説明　(1)キーボードから指定した変数に入力します。

(2)メッセージがある場合、メッセージを表示し、続けて“？”を表示します。

(3)メッセージが省略された場合、“？”のみが表示されます。

(4)データ入力の最後にはEXEキーを押します。

(5)数値変数に文字データを入力しますとエラー(ERR2)となり、ACを押した後に再度“？”を表示してデータ入力を促します。但し、アルファベット1文字や数式を入力した場合、数式の結果が数値のときはその値が代入されます。

(6)データ入力待ちのときにEXEキーだけを押すと、ヌル（何もない）入力となり、変数が数値変数のときはエラー(ERR2)となります。

例

```
10 INPUT A
20 INPUT "B$=",B$
30 INPUT "C$=",C$,"D$=",D$
```

KEY$ (キーダラー) Ⓟ

機能　キーボードから1文字を入力する関数です。

説明
(1)キーボードからのキー入力を1文字分だけ受け入れます。

(2)数字、アルファベット、記号がキー入力できます。

(3)読み込まれたデータは1文字の文字型となります。

(4)"？"は表示せず、入力待ちにもなりませんので、通常はIF文と組み合わせて使います。

※KEY$はKEYと省略することもできます。

例

```
10 PRINT "INPUT<6>";
20 A$=""
30 K$=KEY$
40 IF K$=""THEN 30
50 A$=A$+K$
60 IF LEN(A$)<6 THEN 30
70 PRINT A$
80 END
```

● 6文字をキーボードから受けつけます。

PRINT 〔出力要素〕[{,;}〔出力要素〕]* Ⓟ (プリント)

機能 出力要素を表示します。

パラメータ 出力要素：出力制御関数(CSR)、数式、文字式。

説明 (1)出力要素を表示します。出力制御関数がつく場合は、それによって決められた位置から表示します。

(2)数式、文字式ではその値を表示します。

(3)出力要素が数式の場合は、値の前に符号桁(+,−)がつきますが、+符号は空白として表示されます。

●文字型表示———␣␣␣……␣（出力要素）

●数値型表示———␣␣␣␣……␣（符号 出力要素）

(4)出力要素が数式で仮数部が10桁以上の場合は、11桁目を四捨五入して表示します。また、仮数部以外に符号桁と指数部があるときは指数記号(E)と指数部2桁を表示します。

(5)出力要素と出力要素の区切りは,と;が使え、,のときは前の出力要素を表示後停止し(STOP点灯)、EXEキーにより、一度表示をクリアーしてから、次の出力要素を表示します。区切りが;のときは前の出力要素に続けて次の出力要素を表示します。

(6)出力要素が全て省略されたとき(PRINTのみ)は、表示をクリアーするだけで停止はしません。

(7)MODE 7と押すプリントモードで印字中は、PRINT文を実行しても表示は停止しません。

(8)SET文により数値をフォーマット化することができます。

例

```
10 PRINT 1/3
20 PRINT "A="; A
30 PRINT "SIN 30", SIN 30
40 PRINT "END";
50 PRINT
60 END
```

CSR 出力位置指定 数式 (シーエスアール) Ⓕ

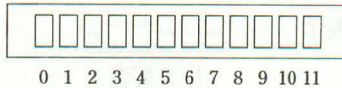
機能 指定した位置から出力要素を表示します。

パラメータ 出力位置指定：数式で、値は小数点以下切り捨てとなります。

0≦指定＜12

説明 (1)PRINT文中で用い、出力要素の出力位置を指定します。

(2)出力位置の与え方は左端を0とします。

0 1 2 3 4 5 6 7 8 9 10 11

例

```
10 FOR I=0 TO 11
20 PRINT CSRI;"A";CSR 11-I;"B"
30 NEXT I
40 END
```

● AとBの文字がEXEキーを押すごとに左右から移動します。

GOTO {分岐先行番号 / ＃プログラムエリア番号} Ⓟ

分岐先行番号: 行番号

＃プログラムエリア番号: 0～9の1文字

（ゴートゥー）

機能 指定された分岐先へ無条件で分岐します。

パラメータ 分岐先行番号：1～9999の行番号（1≦行番号＜10000）
プログラムエリア番号：0～9の1文字

説明 (1)指定された分岐先へ分岐します。

(2)分岐先が行番号の場合は、現在のプログラムエリア内の指定行へ分岐し、プログラムを実行します。分岐先行番号が存在しない場合は、エラー(ERR 4)となります。

(3)分岐先がプログラムエリア番号の場合は、指定されたプログラムエリアへ分岐し、先頭からプログラムを実行します。

※分岐先行番号、プログラムエリア番号に数式も使えます。

例

```
 10 PRINT "START";
 20 GOTO 100
 30 PRINT "LINE 30"
 40 END
100 PRINT "LINE 100"
110 GOTO 30
```

ON 分岐条件 GOTO 〔分岐先〕〔,〔分岐先〕〕* Ⓟ

数式

分岐先は { 分岐先行番号 / #プログラムエリア番号 }

(オン～ゴートゥー)

機能 分岐条件に従って指定された分岐先へ分岐します。

パラメータ 分岐条件：数式で、値は小数以下切り捨てとなります。

分岐先行番号：1～9999の行番号(1≦行番号<10000)

プログラムエリア番号：0～9の1文字

説明 (1)分岐条件の式の値の整数部により分岐します。分岐先は先頭から順に式の値が1の場合、2の場合……と割り当てられます。

ON A GOTO 100(A=1), 200(A=2), 300(A=3), ………

(2)式の値が1より小さいか、または相当する分岐先が書いてないときは分岐せずに、次の文を実行します。

(3)分岐先は一行に納まるまで、いくつでも書けます。

例

```
 10  INPUT A
 20  ON A GOTO 100,200,300
 30  PRINT "OTHER"
 40  GOTO 10
100  PRINT "LINE 100":GOTO 10
200  PRINT "LINE 200":GOTO 10
300  PRINT "LINE 300":GOTO 10
```

● 1～3を入力すると100～300行に分岐し、それ以外では"OTHER"を表示します。

IF 分岐条件 THEN {文〔:文〕* / 分岐先} Ⓟ

比較式

★分岐先は { 分岐先行番号 / #プログラムエリア番号 }

(イフ〜ゼン)

機　能

分岐条件が成立したとき、THEN以降の文を実行します。また THEN以降が分岐先の場合は分岐します。

パラメータ

分岐条件：比較式
分岐先行番号：1〜9999の行番号（1≦行番号＜10000)
プログラムエリア番号：0〜9の1文字

説　明

(1)分岐条件が成立したとき、THEN以降の文を実行または分岐先へ分岐します。

(2)分岐条件が成立しなかったときは、次の行を実行します。

(3)分岐条件は比較式(＝、≠、＜、＞、≦、≧)により判断します。

＝ 左辺と右辺が等しい	≠ 左辺と右辺が等しくない
＜ 左辺より右辺が大きい	＞ 左辺より右辺が小さい
≦ 左辺より右辺が大きいか等しい	≧ 左辺より右辺が小さいか等しい

(4) 2つ以上分岐条件がある場合は、THENの後にIF文を続けることができます。

IF〜THEN IF〜THEN………

※THENの後が文の場合は、THENのかわりに；が使えます。

例

```
10 N=6
20 PRINT CSR N;"↑";
30 K$=KEY$
40 IF K$="4"THEN N=N-1:IF N<0THEN N=0
50 IF K$="6"THEN N=N+1:IF N>11THEN N=11
60 PRINT
70 GOTO 20
```

●"↑"が④キーを押すと左に、⑥キーを押すと右に動きます。

FOR 制御変数名＝初期値(数式) TO 終値(数式)［STEP 刻み幅(数式)］Ⓟ NEXT 制御変数名 （フォー～トゥー～ステップ・ネクスト）

機能 FOR文からNEXT文までの間を制御変数を初期値から終値まで刻み幅で変化させながら繰り返します。

パラメータ 制御変数名：単純変数名で、配列変数は使えません。
初期値：数式
終値：数式
刻み幅：数式。省略したときは1の値

説明 (1)FOR文からNEXT文までの間を制御変数を初期値から終値まで、刻み幅で変化させながら繰り返し、制御変数が終値を超えたとき繰り返しを終了します。

(2)初期値が終値を超えている場合は、FOR～NEXTの間を1度だけ実行します。

(3)刻み幅は負数も使え、省略した場合は1となります。

(4)FOR文とNEXT文は必ず1対1で対応していなければなりません。また、FOR文に対応するNEXT文は、FOR文より後に書きます。

(5)FOR～NEXTのループは次のように入れ子構造にすることができます。

```
10 FOR I=1 TO 10
20 FOR J=11 TO 20
30 PRINT I;":";J
40 NEXT J
50 NEXT I
60 END
```

(6)入れ子構造にすることをネスティングともいい、4重までできます。

(7)FOR～NEXTループを終了したとき、制御変数は終値を超えたときの値となります。

(8)FOR～NEXTループからの外への飛び出しは可能ですが、IF文、GOTO文などでループ内へ飛び込むとエラーになります。なお、ループから飛び出したときも、ループの中であることを記憶していますので、NEXT文で終了させない限りネスティングを重ねていきます。

GOSUB {分岐先行番号 / 行番号 | #プログラムエリア番号 / 0～9の1文字} Ⓟ

(ゴーサブ)

機能 指定された分岐先へサブルーチン分岐します。

パラメータ 分岐先行番号：1～9999の行番号（1≦行番号＜10000）
プログラムエリア番号：0～9の1文字

説明 (1)指定された分岐先へサブルーチン分岐します。サブルーチンからの戻りはRETURNの実行により行なわれます。

(2)サブルーチンの中からさらにサブルーチンを引用することをネスティングといい、8段までできます。

(3)RETURNによりGOSUB文の次の文に戻ります。

(4)GOSUB文によりサブルーチン分岐したときに、IF文やGOTO文などで次の文に戻すと、ネスティングは記憶されたままですので、必ずRETURNで戻してください。

(5)分岐先行番号が存在しない場合はエラー(ERR 4)となります。

※分岐先行番号、プログラムエリア番号に数式も使えます。

例

```
 10 PRINT "MAIN 10"
 20 GOSUB 100
 30 PRINT "MAIN 30"
 40 END
100 PRINT "SUB 100"
110 GOSUB 200
120 RETURN
200 PRINT "SUB 200"
210 RETURN
```

RETURN Ⓟ

(リターン)

機能 サブルーチンから復帰します。

説明 サブルーチンを呼んだ直後の文へ復帰します。

ON 分岐条件/数式 GOSUB〔分岐先〕〔,〔分岐先〕〕*

Ⓟ

★分岐先は {分岐先行番号 / #プログラムエリア番号}

(オン～ゴーサブ)

機　能　分岐条件に従って指定された分岐先のサブルーチンへ分岐します。

パラメータ　分岐条件：数式で、値は小数以下切り捨てとなります。
分岐先行番号：1～9999の行番号（1≦行番号＜10000）
プログラムエリア番号：0～9の1文字

説　明　(1)分岐条件の式の値の整数部によりサブルーチン分岐します。
分岐先は先頭から順に式の値が1の場合、2の場合……と割り当てられます。

ON B GOSUB 1000(B=1), 2000(B=2), 3000(B=3)……

(3)式の値が1より小さいか、または相当する分岐先が書いてないときは分岐せずに、次の文を実行します。

(3)分岐先は一行に納まるまで、いくつでも書けます。

例

```
 10 INPUT A
 20 ON A GOSUB 100,200,300
 30 GOTO 10
100 PRINT "SUB 100":RETURN
200 PRINT "SUB 200":RETURN
300 PRINT "SUB 300":RETURN
```

●1～3を入力すると各々のサブルーチンへ分岐します。

DATA 〔データ〕〔,〔データ〕〕* Ⓟ

定数　定数　(データ)

機能　データを収納します。

パラメータ　データ：文字定数または数値定数

説明
(1)READ文で読み取るデータを書く為に用います。
(2)データは，で区切って複数個書くことができます。
(3)データ文だけを実行しても何もしません。
(4)文字定数の中に，を含むときは、データの両端を " で囲んでください。

```
DATA ABC, DEF, "GHI,JKL",……
     1つ目 2つ目  3つ目
```

(5)データを省略すると長さ0の文字列を表わします。

```
DATA A, ,B → DATA A,"",B
DATA ,     → DATA "",""
DATA       → DATA ""
```

READ 変数名〔,〔変数名〕〕* (リード) Ⓟ

機能 DATAの内容を読み取ります。

パラメータ 変数名：数値変数または文字変数。配列変数も可。

説明
(1)現在指定されているDATA文の中のデータを順に割り当て、指定された変数に代入します。

(2)数値変数には数値型のデータのみ読み取れます。

(3)DATA文の中のデータは、行番号の小さい方から大きい方へ、また同一DATA文中では先頭から順に読まれます。

(4)READ文で必要なデータを読んだ後、次のREAD文で読まれるのは、そのさらに後にあるデータです。

(5)プログラムの動作の初めには、どのデータも指定されません。READ文の最初の実行で、そのREAD文のあるプログラムエリアの先頭のデータが読まれ、以後はこのときのプログラムエリアのデータが順に読まれていきます。

(6)RESTORE文により読み込むデータの指定を変えることができます。

(7)READ文の変数よりDATA文中のデータが少ないときはエラー(ERR4)となります。

(8)DATA文中のデータの先頭にスペースがあるときは読みとばします。

例

```
10 DATA 1,2,3
20 READ A,B
30 PRINT A;B
40 DATA 4,5
50 READ C,D,E
60 PRINT C;D;E
70 END
```

●DATA文から順にデータを読み込み、表示します。

RESTORE 〔行番号〕 Ⓟ
数式　（レストア）

機　能　READ文で読むデータの位置を指定します。

パラメータ　行番号：数式で、値は小数以下を切り捨てとなります。

1≦行番号＜10000

説　明　(1)READ文で読むデータのあるDATA文を指定します。

(2)行番号を省略すると、データの指定を解除します。この後最初に実行するREAD文により、そのREAD文のあるプログラムエリアの先頭にあるデータが指定され、読まれます。

(3)行番号を指定すると、RESTORE文が存在するプログラムエリアの行番号が指定されます。以後のREAD文では、そのときのプログラムエリアのデータが次々に読まれます。

(4)指定された行番号が存在しないときや指定された行番号以降にDATA文が存在しないときは、エラー(ERR4)となります。

例

```
10 DATA 1,2,3
20 DATA 4,5
30 READ A,B,C,D,E
40 RESTORE 10
50 READ F,G
60 RESTORE 20
70 READ H,I
80 PRINT A;B;C;D;E;F;G;H;I
90 END
```

PUT 〔"ファイル名"〕変数1〔，変数2〕 Ⓐ

文字列　（プット）

機　能　カセットテープにデータを記録します。

パラメータ　ファイル名：1～8文字の文字列。省略可。

変数1，変数2：記録する変数範囲の指定

説　明　(1)カセットテープに変数の内容を記録します。

(2)変数の指定は次のように書きます。

PUT　A……………………変数Aの内容
PUT　A,Z…………………変数A～Zの内容
PUT　A,A(100)…………変数A～A(100)の内容
PUT　$,D,W……………専用文字変数$とD～Wの内容

専用文字変数$の内容を記録する場合は$を最初に書き込みます。

(3)マニュアルでもプログラム中に書き込んでも使えます。

GET 〔"ファイル名"〕変数1〔，変数2〕 Ⓐ

文字列　（ゲット）

機　能　カセットテープに記録されているデータを変数に読み込みます。

パラメータ　ファイル名：1～8文字の文字列。省略可。

変数1，変数2：読み込む変数の指定

説　明　(1)カセットテープに記録されているデータを指定された変数に読み込みます。

(2)変数の指定は次のように書きます。

GET　A……………………変数Aに読み込む
GET　A,Z…………………変数A～Zに読み込む
GET　A,A(100)…………変数A～A(100)に読み込む
GET　$,D,W……………専用文字変数$とD～Wに読み込む

(3)PUTにより記録した変数名とGETにより読み込む変数名は一致していなくてもかまいません。

(4)読み込もうとする変数より記録されているデータが少ない場合は、記録されているデータだけ、先頭の変数から順に読み込みます。

(5)マニュアルでもプログラム中に書き込んでも使えます。

BEEP 〔{0 / 1}〕 Ⓐ

(ビープ)

機　能　ビープ音を発生させます。

パラメータ　0：低音。

1：高音。

省略した場合は0と同じ。

説　明　(1)低い音と高い音のビープ音を発生させます。

(2)マニュアルでも使えます。

例

```
 10 $="ABCDEFGHIJKLMNOPQRSTUVWXYZ":N=0
 20 FOR I=1 TO 10
 30 A$=MID$( RAN#*26+1,1 )
 40 PRINT CSR4;"<";A$;">";
 50 FOR J=1 TO 30
 60 K$=KEY$:IF K$≠""THEN80
 70 NEXT J
 80 IF K$=A$ THEN BEEP 1:N=N+1:GOTO 100
 90 BEEP 0
100 PRINT:NEXT I
110 PRINT N;
120 IF N>10 THEN END
130 FOR I=1 TO 10
140 BEEP 0:BEEP 1
150 NEXT I
```

●表示された文字に対応するアルファベットキーを押します。

DEFM 〔増設メモリー数 / 数式〕 Ⓐ

(ディーイーエフエム)

機能 メモリーの増設をします。

パラメータ 増設メモリー数：数式で、値は小数以下切り捨てとなります。
省略可。0 ≦増設メモリー数≦196。(OR-1Ⓔ装着時)

説明
(1)メモリー(変数エリア)の増設を行ないます。

(2)増設メモリー数は1個単位で残りステップ数に応じて任意に指定できます。

(3)メモリー1つ増設するごとに8ステップ必要となりますので、プログラムエリアのステップ数は減少します。

(4)配列変数を使用し、データメモリーを多く必要とするときに使用します。

(5)増設メモリー数を省略した場合は、現在設定されているメモリー数を表示します。

(6)マニュアルでもプログラム中に書き込んでも使え、マニュアルで実行した場合は新しい設定状態(増設メモリー数+基本メモリー数26)を表示します。プログラム中に書き込んで実行した場合は新しい設定状態は表示されません。

(7)残りステップ数よりも多くのメモリーを増設しようとした場合は、エラー(ERR1)となります。

(8)増設したメモリーをクリアーにして、最初の26個に戻すには、DEFM 0を指定します。

例

入力	表示
DEFM 10 EXE	***VAR:36
DEFM EXE	***VAR:36

```
10 DEFM 10
20 FOR I=1 TO 10
30 INPUT Z(I)
40 NEXT I
  :
```

MODE 数式 (モード) Ⓟ

機　能　計算機の状態を設定します。

パラメータ　数式：値は小数以下切り捨てとなります。

4≦数式＜9

説　明　(1)数式の値により角度単位やプリントモードの設定・解除を設定します。

(2)設定は次の通りです。

MODE4………角度単位を度に設定します。
MODE5………角度単位をラジアンに設定します。
MODE6………角度単位をグレードに設定します。
MODE7………"PRT"を表示し、プリントモードに設定します。
MODE8………プリントモードを解除します。

(3)これはMODEキーによる設定と同じですが、RUNモードやWRTモードの設定はできません。

(4)入力方法はMODEキーではなく、MODEとアルファベットキーを使います。

例

```
10 MODE 4
20 A=SIN 30
30 MODE 7
40 PRINT A
50 MODE 8
60 END
```

SET {Fn / En / N}　★nは0～9の整数　Ⓐ (セット)

機能 数値データの出力形式(フオーマット)を指定します。

パラメータ
Fn：小数点以下指定。
En：有効桁数指定。
N ：指定解除。

説明
(1)数値データを出力するときの、小数点以下の桁数や有効桁数の指定を行ないます。

(2)小数点以下指定(Fn)では、小数点以下の桁数を0から9桁まで指定します。

(3)有効桁数指定(En)では、有効桁数を1から10桁まで指定します。なお、"**SET E0**"としたときは10桁指定となります。

(4)"**SET N**"では両指定を解除します。

(5)マニュアルでもプログラム中に書き込んでも使えます。

例
```
10 INPUT N
20 SET F5:PRINT N
30 SET E5:PRINT N
40 SET N:GOTO 10
```

LEN (単純文字変数)　(レングス)　Ⓕ

機　能　与えられた単純文字変数の中の文字数を値とする関数です。

パラメータ　単純文字変数：配列変数は使えません。

説　明　(1)単純文字変数の中の文字数を数える関数です。

(2)使える文字変数は単純文字変数(A$, Y$等)で、配列文字変数(B$(3)等)は使えません。

例

```
10 INPUT A$
20 PRINT LEN(A$)
30 GOTO 10
```

MID$ (位置〔, 文字数〕)

位置：数式　文字数：数式

Ⓕ

(ミッドダラー)

機能

専用文字変数($)に記憶されている文字列の、指定した位置から指定文字数を取り出す関数です。

パラメータ

位置　：数式で、値は小数点以下を切り捨てます。

1≦位置<101

文字数：数式で、値は小数点以下を切り捨てます。

1≦文字数<101

省略すると、位置以降の全てが指定されます。

説明

(1)専用文字変数($)に記憶されている文字列の、指定した位置から指定した文字数分の文字を取り出す関数です。

(2)位置が文字列の範囲を超えて指定されたとき（文字数より位置の方が大きいとき)は、ヌル(何もない)を与えます。

(3)文字列の、指定した位置以降の長さが指定した文字数より小さいときは、指定した位置以降の文字列全部を取り出します。

※MID $はMIDと省略することができます。

例

```
10 $="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
20 INPUT M,N
30 PRINT MID$(M,N)
40 END
```

VAL（単純文字変数） Ⓕ

（バリュー）

機能　単純文字変数内の数字を数値に変換する関数です。

パラメータ　単純文字変数：配列変数は使えません。

説明　(1)単純文字変数内の数字を数値に変換します。

(2)文字変数の内容が数字のみ（+、−、・、E、E⁻を含む）の場合は、そのまま数値に変換します。

A＄＝"−12.3"のとき、VAL（A＄）→−12.3

(3)文字変数の内容が数字以外で始まっている場合は、エラーとなります。

A＄＝"A45"のとき、VAL（A＄）→エラー（ERR 2）

(4)文字変数の内容が数字で始まっているが、途中に数字以外が入っている場合は、最初の数字部分を数値に変換します。

A＄＝"78A9"のとき、VAL（A＄）→78

例

```
10 INPUT A$
20 PRINT VAL(A$)
30 END
```

STR$（数式） Ⓕ

（ストリングダラー）

機能　数式の値を文字（数字）に変換します。

パラメータ　数式：数値、計算式、数値変数、配列数値変数

説明　(1)数式の値を文字に変換します。

(2)数式が計算式の場合は、計算結果を文字に変換します。

(3)数式が正の場合は、符号桁は削除され、数字だけとなります。

例

```
10 PRINT STR$(123)
20 PRINT STR$(45+78)
30 A=963
40 PRINT STR$(A)
50 END
```

数値関数

SIN 引数/数式　COS 引数/数式　TAN 引数/数式 Ⓕ

機　　能　与えられた引数に対する三角関数の値を与える関数です。

パラメータ　引数：数式。

−1440°＜引数＜1440°　（度）

−8π＜引数＜8π（ラジアン）

−1600＜引数＜1600（グレード）

但し、TANにおいては |引数|＝(2n -1)＊(1 直角)を除く

$1\text{直角}=90^\circ=\frac{\pi}{2}\text{rad}=100\text{grad}$

説　　明　(1)与えられた引数に対する三角関数の値を与える関数です。

(2)値は角度単位の設定(MODEキー、モードコマンド)に従います。

ASN 引数/数式　ACS 引数/数式　ATN 引数/数式 Ⓕ

機　　能　与えられた引数に対する逆三角関数の値を与える関数です。

パラメータ　引数：数式。ASN、ACSは-1≦引数≦1。

説　　明　(1)与えられた引数に対する角度を与える逆三角関数です。

(2)値は角度単位の設定(MODEキー、モードコマンド)に従います。

(3)関数の値は以下の範囲で与えられます。

−90°≦ASN　X　≦90°

0°≦ACS　X　≦180°

−90°≦ATN　X　≦90°

LOG 引数/数式　LN 引数/数式 Ⓕ

機　能	対数関数の値を与える関数です。
パラメータ	引数：数式。 0＜引数。
説　明	対数関数の値を与えます。

●LOG　常用対数関数　$\log_{10} x$、$\log x$
●LN　自然対数関数　$\log_e x$、$\ln x$

EXP 引数/数式 Ⓕ

機　能	指数関数の値を与える関数です。
パラメータ	引数：数式。 -10^{100}＜引数≦230.2585092
説　明	指数関数の値を与える関数です。

EXP　e^x

SQR 引数/数式 Ⓕ

機　能	引数の平方根を与える関数です。
パラメータ	引数：数式。 0≦引数。
説　明	引数の平方根を与える関数です。

SQR　$\sqrt{x}$

ABS 引数
数式 Ⓕ

機能　引数の絶対値を与える関数です。

パラメータ　引数：数式。

説明　引数の絶対値を与える関数です。

ABS　　$|x|$

SGN 引数
数式 Ⓕ

機能　引数の符号に応じた値を与える関数です。

パラメータ　引数：数式。

説明　引数の符号に応じた値を与える関数です。

引数が正の場合、1

引数が0の場合、0

引数が負の場合、-1

INT 引数
数式 Ⓕ

機能　引数を越えない最大の整数を与える関数です。

パラメータ　引数：数式。

説明　引数を越えない最大の整数を与える関数です。

INT 12.56→12

INT −78.1→−79

FRAC 引数（数式） Ⓕ

機能	引数の小数部を値とする関数です。
パラメータ	引数：数式。
説明	引数の小数部を値とする関数です。符号は引数の符号と一致します。

RND (引数（数式）,桁位置（数式）) Ⓕ

機能　引数を指定した桁で四捨五入した値を与える関数です。

パラメータ　引数：数式。

桁位置：数式。値は小数点以下切り捨てとなります。

－100＜桁指定＜100

説明　(1)引数を指定した桁で四捨五入した値を与える関数です。

(2)桁指定は10の桁位置乗の位置を四捨五入します。

小数点以下3桁目を四捨五入→RND(x, －3)

100の位を四捨五入――――→RND(x, 2)

RAN＃ Ⓕ

機能　0から1の間の乱数を与える関数です。

説明　(1) 0から1の間の乱数を与える関数です。0＜乱数＜1

(2)乱数は小数点以下10桁です。

例　0～9の1桁の乱数を作る

```
INT(RAN#*10)
```

1～5の1桁の乱数を作る

```
INT(RAN#*5+1)
```

10～99の2桁の乱数を作る

```
INT(RAN#*90+10)
```

DEG (度〔,分〔,秒〕〕) Ⓕ

（度：数式、分：数式、秒：数式）

機能　60進数を10進数に変換します。

パラメータ　度：数式。

分：数式。

秒：数式。

|DEG (度, 分, 秒)| ＜ 10^{100}

説明　度・分・秒で示される60進数を10進数に変換します。

例　DEG(12,34,56) EXE　　12.58222222

```
10 INPUT A,B,C
20 PRINT DEG(A,B,C)
30 END
```

DMS$ (引　数 / 数式) Ⓕ

機　能　10進数を60進数に変換します。

パラメータ　引数：数式。|数式|＜10^{100}

説　明　(1)10進数を60進数に変換します。

(2)変換された結果は、文字列として与えられます。

例　DMS$(45.678) EXE　　45°40'40.8

```
10 INPUT A
20 $=DMS$(A)
30 PRINT$
40 END
```

データバンク用コマンド

NEW# Ⓜ

（ニュークロスハッチ）

機能 データバンクデータの消去

説明
(1)データバンクに記憶されているデータを全て消します。
(2)パスワード設定中は実行できません。
(3)WRTモードでのみ実行できます。

例
MODE 1
NEW# EXE
MODE 0

LIST# Ⓜ

（リストクロスハッチ）

機能 データバンクデータを全て表示します。

説明
(1)データバンクに記憶されているデータを記録された順に表示します。
(2)表示される内容はレコードナンバー（記録された順番）とメモデータです。
(3)データバンクデータは順に自動的に表示されますので、止めたいときはSTOPキーを押します。再び次を表示させたいときはEXEキーを押します。
(4)プリントモード（MODE 7と押す）では表示は止まらず、順次早い速度で表示します。
(5)パスワード設定中は実行できません。
(6)メモインモード（MODE 9と押す）では実行できません。

例 LIST# EXE

SAVE# 〔"ファイル名"〕 Ⓜ
文字列 （セーブクロスハッチ）

機能	データバンクデータをカセットテープに記録します。
パラメータ	ファイル名：1～8文字の文字列。省略可。
説明	(1)データバンクに記憶されているデータを全て、カセットテープに記録します。 (2)SAVE、SAVE ALLではデータバンクデータを記録しませんので、メモデータは必ずこのSAVE#で記録してください。 (3)パスワードが設定されている場合は、パスワードをつけて記録しますので、LOAD#コマンドにより読み込んだときに同じパスワードがつきます。 (4)メモインモードでは実行できません。
例	SAVE# EXE SAVE#"CASIO" EXE

LOAD# 〔"ファイル名"〕 Ⓜ
文字列 （ロードクロスハッチ）

機能	データバンクデータをカセットテープから読み込みます。
パラメータ	ファイル名：1～8文字の文字列。省略可。
説明	(1)カセットテープに記録されているデータバンクデータを読み込みます。 (2)パスワードつきで記録されたデータバンクデータを読み込みますと、記録したときと同じパスワードが設定されます。 (3)すでにデータが入っている場合は、前のデータをクリアーしてから新たなデータを読み込みます。 (4)メモインモードでは実行できません。
例	LOAD# EXE LOAD#"CASIO" EXE

READ# 変数名〔,変数名〕* (リードクロスハッチ) Ⓟ

機能 データバンクデータを読み取ります。

パラメータ 変数名:数値変数または文字変数。配列変数も可。

説明 (1)データバンクに記憶されているデータを、順に変数に読み込みます。

(2)数値変数には数値型データのみ読み取られ、文字型データの場合はエラー(ERR2)となります。

(3)READ#で必要なデータを読んだ後、次のREAD#で読み込まれるのは、さらにその後にあるメモデータです。

(4)データバンクデータが,で区切られているときは、,ごとにデータを読みます。

例)データ

No.1 A,X,Y
No.2 B,Z
No.3 C

⬇

読み込む順番
A→X→Y→B→Z→C

(5)読むべきデータがない場合はエラー(ERR4)となります。

(6)RESTORE#(125ページ参照)により読み込むデータの順番を変更することができます。

(7)データバンクデータの先頭にスペースがある場合は、スペースを読み飛ばします。

(8)データバンクデータが" "(ダブルクォーテーション)で囲まれているときは、" "の中の文字列を読み込みます。

例

〈メモデータ〉

No.1 1,2,3
No.2 4,5,6
No.3 7,8,9
No.4 10,

〈プログラム〉

```
10 A=0
20 READ#$
30 IF $="" THEN 60
40 A=A+VAL($)
50 GOTO 20
60 PRINT "Σx=";A
70 END
```

●電子メモから数値データを読み込み、合計を求めます。

RESTORE# 〔"検索文字列" / 文字式〔, [{0 / 1}]〔, {行番号 / #プログラムエリア番号}〕〕〕 Ⓟ

（レストアクロスハッチ）

機　能　データバンクデータを検索し、READ＃で読むメモデータの順番を設定します。

パラメータ

検索文字列　　　　　：文字式。文字列の場合は" "で囲みます。
行番号　　　　　　　：数式。0＜行番号＜10000
プログラムエリア番号：数式。0≦プログラムエリア番号＜10

説　明

⑴データバンクデータを検索し、次のREAD＃で読むデータの順番を設定します。

⑵パラメータとデータ検索の関係は次のようになっています。

①RESTORE＃

検索文字列以降が省略された場合には、次のREAD＃で読むデータは先頭からとなります。

②RESTORE＃"検索文字列"

検索文字列を先頭に含むデータを検索し、該当するデータが次のREAD＃で読まれます。

③RESTORE＃"検索文字列", {0 / 1}

0が指定された場合は②(省略)と同じになります。

1が指定された場合は、検索したデータを含む行の先頭データが、次のREAD＃で読まれます。

④RESTORE＃"検索文字列", [{0 / 1}], {行番号 / #プログラムエリア番号}

検索を実行して該当するメモデータがない場合、指定された行番号またはプログラムエリアに分岐します。

※②および③において、該当するメモデータがない場合はエラー(ERR4)となります。

※④で、分岐先行番号が存在しないときや、分岐先プログラムエリアにプログラムが存在しないときはエラー(ERR4)となります。

例

〈メモデータ〉

No. 1 SUZUKI,347-4811,SHINJUKU
No. 2 KOMATSU,045-211-0821,YOKOHAMA
No. 3 SATO,06-314-2681,OSAKA
No. 4 SHIMIZU,075-351-1161,KYOTO

〈プログラム〉

```
 10 RESTORE#
 20 READ# $
 30 PRINT $
 40 RESTORE#"K"
 50 READ# $
 60 PRINT $
 70 RESTORE#"KY",1
 80 READ# $
 90 PRINT $
100 RESTORE#"AA",1,200
110 READ# $
120 PRINT $
130 END
200 PRINT"MEMO END"
210 END
```

- 10〜30：先頭に記憶されているデータを表示します。
- 40〜60：頭文字がKであるデータを表示します。
- 70〜90：先頭2文字がKYであるデータを検索し、そのデータのある行の先頭データを表示します。
- 100〜120：先頭2文字がAAであるデータがないときは行番号200へ分岐します。

操作	表示
RUN EXE	SUZUKI
EXE	KOMATSU
EXE	SHIMIZU
EXE	MEMO END

WRITE# 〔メモデータ〔, メモデータ〕*〕 Ⓟ
式　式

(ライトクロスハッチ)

機能 データバンクデータの書きかえまたは削除をします。

パラメータ メモデータ：数式および文字式。文字列の場合は" "で囲みます。

説明
(1)現在RESTORE#等で指定されているレコードナンバーのデータバンクエリアにメモデータを書き込みます。
(2)該当レコードのデータの有無にかかわらず、新たに書き込まれます。
(3)メモデータが全て省略されて実行されたときは、そのレコードにあったメモデータを消去します。
(4)メモデータが複数個あるときは、,で区切って書くことができます。このときのデータは,といっしょに一行に書き込まれます。
(5)WRITE#で書き込まれた後のレコードナンバー指定は、書き込まれた次のメモデータとなります。

例

```
 10 REM WRITE
 20 RESTORE#           ┐
 30 WRITE#"A,B,C"      ┘── 新たなデータを書きます。
 40 RESTORE#
 50 FOR I=1 TO 3
 60 READ# $:PRINT $;
 70 NEXT I
 80 PRINT" "
 90 REM CHANGE
100 RESTORE#           ┐
110 FOR I=1 TO 3       ├── データを書きかえる。
120 WRITE# STR$(I)     ┘
130 NEXT I
140 RESTORE#
150 FOR I=1 TO 3
```

```
160 READ# $: PRINT $;
170 NEXT I
180 PRINT" "
190 REM CLEAR
200 RESTORE#
210 WRITE#          ————————データの消去
220 RESTORE#
230 READ# $
```

操　作	表　示
RUN EXE	ABC
EXE	123
EXE	ERR4 P0-230

↓
データ消去のためデータ不足

第6章

PB-110を活用しよう

この章ではPB-110をもっと活用していただくために、すぐに使えるサブルーチン集や豊富にあるPB-100シリーズのソフトを利用する方法等について説明しております。BASICプログラムについて多少なりとも理解されましたなら、この章をお読みになってよりいっそう活用してみてください。

6−1 楽しいサブルーチン集

ここではゲームに使うと楽しめるプログラムを入力、出力、その他の3つに分類してサブルーチンとして記載しています。
これ等のサブルーチンを継ぎ合わせるだけでも、楽しいゲームができると思います。
実際の使われ方は135ページまたは第7章のライブラリー編を参照してください。

■入力サブルーチン

①プログラム実行後、キーをはなすまで先に進まない。

```
1000 IF KEY$≠"" THEN 1000
```

※KEY$関数とIF文により、キーが押されているかを判断し、押されていれば行番号1000を繰り返します。

②設定した範囲の整数だけを入力する。

```
1000 INPUT "LEVEL(1-9)",N
1010 N=INT N
1020 IF N<1 THEN 1000
1030 IF N>9 THEN 1000
```

※設定した範囲（ここでは1から9までの整数）をIF文により判断させ、範囲内でなければ再入力とします。

③一定の文字数だけ入力を受けつける。

```
1000 A$=""
1010 FOR B=1 TO 7
1020 $=KEY$:IF $="" THEN 1020
1030 A$=A$+$:BEEP
1040 NEXT B
```

※FOR・NEXT文により7文字分繰り返します。行番号1020のKEY$関数とIF文により、キーを押していなければ行番号1020を繰り返し、キーを押したときだけ行番号1030で文字変数A$に記憶させます。

④1文字分のキー入力が0から9の数字キーでなければ再入力します。

```
1000 $=KEY$
1010 IF $<"0" THEN 1000
1020 IF $>"9" THEN 1000
```

※KEY$関数で入力された1文字が、"0"より小さいか"9"より大きければ戻ります。文字もキャラクター表(165ページ)の順で大小比較ができます。
"0"のかわりに"A"とし、"9"のかわりに"Z"とすれば、アルファベット26文字のどれかのキーを押したときだけ先に進むようになります。

⑤④キーを押せば表示位置を左に、⑥キーを押せば表示位置を右に移動させる。

```
1000 $=KEY$
1010 IF $="4" THEN X=X-1:IF X<0 THEN X=0
1020 IF $="6" THEN X=X+1:IF X>11 THEN X=11
```

※表示位置の指定を変数Xで行なうとします。④キーが押されれば行番号1010でXから1を減じ、⑥キーが押されれば行番行1020でXに1を加えます。行番号1010と1020の後半のIF文は、変数Xの値が表示範囲(0～11)をこえないようにチェックしています。

⑥⑧キーを押すと音が早く鳴り、②キーを押すと音がゆっくりと鳴る。

```
1000 N=20
1010 $=KEY$
1020 IF $="8" THEN N=N-2
1030 IF $="2" THEN N=N+2
1040 FOR I=1 TO N:NEXT I
1050 BEEP 1
1060 GOTO 1000
```

※⑧キーと②キーが押されるごとに行番号1040のFOR・NEXTループの終値を変えて、時間の間隔を変化させます。

⑦キーを押している時間をはかる。

```
1000 BEEP 0:N=0
1010 IF KEY$="" THEN BEEP 1:RETURN
1020 N=N+1
1030 GOTO 1010
```

※低音を鳴らした後、キーが押されていれば行番号1020でカウントを繰り返し、キーをはなしたときに高音を鳴らして戻ります。

■出力サブルーチン

①一定時間キャラクターを表示後、表示をクリアーする。

```
1000 PRINT "♠";
1010 FOR V=1 TO 100
1020 NEXT V
1030 PRINT
```

※行番号1000で"♠"を表示させた後、行番号1030で表示をクリアーにします。行番号1010のFOR文中の終値「100」を変えることにより、表示している時間が変わります。

②文字を１文字づつ音とともに表示する。

```
1000 $="PB-110"
1010 FOR V=1 TO LEN($)
1020 PRINT MID$(V,1);
1030 BEEP 1
1040 NEXT V
```

※専用文字変数($)に記憶された文字列を、FOR・NEXT文で１文字づつ順に取り出して表示させます。表示されるときにそのつどBEEP音を鳴らします。

③キャラクターが左から右へ音とともに移動する。

```
1000 FOR A=0 TO 11
1010 PRINT CSRA;"*";
1020 BEEP 0
1030 PRINT CSRA;" ";
1040 NEXT A
```

※表示位置はFOR・NEXT文を使い０から11までとし、行番号1010でまず"*"を表示させ、音を鳴らしたあとに行番号1030で表示されていた"*"を消します。

④２つのキャラクターを同じ位置に交互に表示させ、動きを作る。

```
1000 FOR V=1 TO 10
1010 PRINT CSRX;"*";:BEEP 1
1020 PRINT CSRX;"+";:BEEP 0
1030 NEXT V
```

※表示位置はあらかじめ変数Ｘに数値を代入しておき、FOR・NEXT文により10回"*"と"+"を交互に表示させます。

⑤キャラクターが転がりながら左から右に移動する。

```
1000 A$="+":B$="X"
1010 FOR X=0 TO 11
1020 PRINT CSRX;A$;:BEEP 0
1030 PRINT CSRX;" ";:BEEP 1
1040 C$=A$:A$=B$:B$=C$
1050 NEXT X
```

※FOR・NEXT文により"+"と"X"を交互に移動させながら表示させます。
行番号1040は入れ替えで、A$の内容とB$の内容をとり替えます。

⑥同じ位置で数字がカウントアップする。

```
1000 FOR A=1 TO 100
1010 $=STR$(1000+A)
1020 PRINT CSR0;MID$(2);
1030 NEXT A
```

※行番号1010で4桁の数字に変換し、行番号1020で2文字目以降を表示させます。

⑦背景となるキャラクターを動かします。

```
1000 $="♠  ♠  ♠  ♠  ♠"
```
（♠と♠の間：スペース2個分）

```
1010 FOR N=1 TO 3
1020 PRINT CSR0;MID$(N,12);
1030 PRINT CSR2;"*";
1040 BEEP 0
1050 NEXT N
1060 GOTO 1010
```

※専用文字変数($)に背景となるキャラクターを記憶させ、表示させる位置を変えながら表示させます。

⑧9桁以内の整数に3桁位取(,)をつける。

```
 100 INPUT A:A=INT A:IF ABS A>1E9
     THEN 100
 110 GOSUB 1000
  ⁝
 900 END
1000 $=STR$(A):B=LEN($)
1010 PRINT "¥";
1020 FOR I=1 TO B
1030 PRINT MID$(I,1);
1040 IF I≠B THEN IF SGN A+I≠0 THEN IF
     FRAC((B-I)/3)=0 THEN PRINT ",";
1050 NEXT I
1060 PRINT ""
1070 RETURN
```

※行番号1000からのサブルーチンにジャンプする前に、変数Aに数値を入力しておきます。サブルーチン内では、数値を文字化して専用文字変数($)に記憶させ、桁数を変数Bに記憶させます。まず行番号1010で「¥」を表示させ、次に最下桁から数えて3桁目の前に入るよう「,」を表示させます。行番号1060のPRINT文は、表示を停止させるためで、この文がないとプログラムは次に進み、もし続けてPRINT文がありますと、続けて表示してしまうからです。

■その他

①乱数により0から9までの1桁の整数を作る。

```
1000 R=INT(RAN#*10)
```

※乱数発生関数(RAN#)により求められる0<乱数<1の乱数を10倍し、0<乱数<10の数値を求めたあとその整数部を取り出します。

②乱数により1から20までの整数を3個作る。

```
1000 FOR D=0 TO 2
1010 A(D)=INT(RAN#*20+1)
1020 NEXT D
```

※①と同様に、乱数を20倍して0<乱数<20を求め、1を加えることにより、1<乱数<21の数値を求めて整数部を取り出します。こうしてできた整数はA(0)〔=A〕、A(1)〔=B〕、A(2)〔=C〕に代入されます。

③乱数によりトランプのマーク1つと1から13の数字を選び出す。

```
1000 $="♠♥♦♣"
1010 A=INT(RAN#*4+1)
1020 B$=MID$(A,1)
1030 C=INT(RAN#*13+1)
```

※行番号1000で専用文字変数($)にトランプのマーク4つを記憶させ、行番号1010で1～4の数値を作り、MID$関数により専用文字変数に記憶されているマークの中から1つを取り出します。行番号1030では乱数により1～13の数値を作ります。

④キャラクターを左右に動かしますが、左端に行くと右端から現われ、右端に行くと左端から現われるようにする。

```
1000 X=5
1010 PRINT CSRX;"*";
1020 $=KEY$
1030 IF $="4" THEN X=X-1:BEEP 1
     :IF X<0 THEN X=11
1040 IF $="6" THEN X=X+1:BEEP 1
     :IF X>11 THEN X=0
1050 PRINT
1060 GOTO 1010
```

※行番号1020で押されたキーが[4]であれば行番号1030のIF文により表示位置を左にしますが、左端より出てしまうとき(X<0)には表示位置を11(右端)にします。[6]キーが押されたときも同様に表示位置を右にし、右端より出るとき(X>11)には表示位置を0(左端)にします。

⑤2つのキャラクターの位置が重なっているかを判断する。

```
1000 X=3:Y=8
1010 PRINT CSRX;"*";CSRY;"#";
1020 IF X=Y THEN RETURN
1030 $=KEY$
1040 IF $="4" THEN X=X-1:BEEP 1
     :IF X<0 THEN X=11
1050 IF $="6" THEN X=X+1:BEEP 1
     :IF X>11 THEN X=0
1060 PRINT:GOTO 1010
```

※基本的には④と同じですが、表示するキャラクターが2つあり、表示位置を変数X・Yで指定しています。この2つのキャラクターが重なるということは変数XとYの値が等しいときですので、行番号1020のIF文で判断させます。

⑥2つのキャラクターを表示させ、1つのキャラクターにもう1つのキャラクターを追いかけさせる。

```
1000 X=INT(RAN#*12)
1010 Y=INT(RAN#*12)
1020 PRINT:PRINT CSRX;"*";CSRY;"#";
1030 IF X=Y THEN BEEP 1:GOTO 1000
1040 IF X>Y THEN Y=Y+1
1050 IF X<Y THEN Y=Y-1
1060 GOTO 1020
```

※ここでは行番号1000と1010で2つの表示位置を乱数により作っています。
行番号1030ではIF文により2つの位置が一致しているかを判断します。
行番号1040と1050では変数Yより変数Xが大きいか小さいかを判断し、「#」の表示位置を左右に移動させます。

〈サブルーチン集使用例〉

```
10 IF KEY$="" THEN 10                      ——入力の①
20 X=5
30 Y=INT(RAN#*12)                          ——その他の①
40 PRINT:PRINT CSRX;"*";CSRY;"#";  ┐
50 IF X=Y THEN 100                  │      ——その他の⑤
60 $=KEY$                           │
70 IF $="4" THEN X=X-1:             ├      ——入力の⑤
   BEEP 0:IF X<0 THEN X=0           │
80 IF $="6" THEN X=X+1:             │
   BEEP 0:IF X>11 THEN X=11         ┘
```

```
 90 GOTO 40
100 FOR I=1 TO 10
110 PRINT CSRX;"*";:BEEP 1
120 PRINT CSRY;"+";:BEEP 0
130 NEXT I
140 GOTO 20
```
（100〜130行 ―― 出力の④）

このプログラムはプログラムリストの右側に書かれている例を集めたものです。
これだけでは完全なゲームとはいえませんが、けっこう楽しめます。
このように、サブルーチンの例をうまく組み合わせ、また応用して楽しいゲームを作ってみてください。

6−2 PB-100のプログラムを使う

PB-100シリーズ(PB-100、PB-200、PB-300、FX-700P、FX-802P)で作られたプログラムは、ビジネスをはじめゲームまでたくさんあり、ライブラリーも出版されています。
本機もシリーズの1つとして、この豊富なプログラムを利用できますが、本機にはPB-100シリーズ以上のコマンドを持っていますので、もっと便利な使い方もできます。ここでは、PB-100シリーズにより作られたプログラムを、本機で使ってみましょう。

■相異点

本機とPB-100シリーズの使っているBASIC言語は、ほとんど共通していますが、本機の方が多くの命令を持っていますし、一部異なる点もあります。

●追加命令

PASS（プログラム保護）
BEEP（ブザー音）
READ（DATA文よりデータを読み込む）
DATA（データを書いておく）
RESTORE（読み込むデータの指定）
ON~GOTO（GOTO文の間接指定）
ON~GOSUB（GOSUB文の間接指定）
REM（注釈文）

●追加関数

DEG（60進数→10進数変換）
DMS$（10進数→60進数変換）
STR$（数値を数字に変換）

●変更命令

本機	PB-100シリーズ
NEW(NEW ALL)	CLEAR(CLEAR A)
CLEAR	VAC
IF~THEN	IF~;
SAVE ALL	SAVE A
LOAD ALL	LOAD A
VERIFY	VER
DEFM(プログラム書き込み可)	DEFM(マニュアルのみ可)

●**変更関数**

本機	PB-100シリーズ
KEY\$ MID\$	KEY MID

以上のような相異点がありますが、PB-100シリーズで作られたプログラムは、**原則としてそのまま使えます。**
実は、本機にはPB-100シリーズで作られたプログラムを判別する機能を持っています。ただし、使いやすくするためや、後からの見直しを便利にするために、本機用に手直しした方が良いと思います。

例)
PB-100シリーズ用プログラム

```
10 VAC
20 FOR A=1 TO 20
30 INPUT Z(A)
40 IF Z(A)>80;B=B+1:GOTO 90
50 IF Z(A)<60;C=C+1:GOTO 90
60 IF Z(A)>40;D=D+1:GOTO 90
70 IF Z(A)>20;E=E+1:GOTO 90
80 F=F+1
90 NEXT A
      :
      :
```

この例はデータを入力して、データの大きさに応じて振り分ける部分です。このようなプログラムでも、そのまま使えますが、本機用のプログラムにするには、次の点を直してください。
行番号10の"VAC"は"CLEAR"にする

```
10 CLEAR
```

行番号40から70の";"は"THEN"にする。

```
40 IF Z(A)>80 THEN B=B+1:GOTO 90
```

(以下同じ)

このプログラムでは変数の増設が必要ですので、PB-100シリーズではマニュアルで実行していたDEFM命令を、プログラムの先頭に書き込みます。

```
5 DEFM 20
```

例2)

PB-100シリーズ用プログラム

```
10 INPUT "I=1/O=2/P=3",N
20 IF N<1 THEN 10
30 IF N>3 THEN 10
40 GOTO N*100
      :
      :
```

このプログラムは作業に合わせて分岐先を振り分けるプログラムですが、これを本機に合わせるには、ON～GOTO文を用いて、次のように変更します。

```
10 INPUT "I=1/O=2/P=3",N
20 ON N GOTO 100,200,300
30 GOTO 10
      :
      :
```

このようにON～GOTO文を使うことによりスッキリできますし、判断によりデータNをチェックする必要がなくなります。

以上の内容に注意すれば、今迄作られているPB-100シリーズ用のプログラムがより有効に使えます。

すぐにでも色々なプログラムを使ってみたい方は、市販されているPB-100シリーズ用のプログラム集を利用できますので、とても便利です。

なお、PB-100シリーズで作成され、テープに記録されているプログラムやデータは、そのまま本機で読み込むことができますが、本機で作成され、テープに記録されているプログラムやデータは、他のPB-100シリーズで読み込めないものもあります。これ等の関係を次に示しますので、注意してお使いください。

本機→PB-400, FX-710P

SAVE ＼ LOAD	PF	AF	MF	パスワードつき PF	パスワードつき AF	パスワードつき MF
LOAD	○		／	○		／
LOAD ALL		○	／		○	／

本機→PB-100, PB-200, PB-300, FX-700P, FX-802P

SAVE ＼ LOAD	PF	AF	MF	パスワードつき		
				PF	AF	MF
LOAD	○		／	／	／	／
LOAD ALL		○	／	／	／	／

○：読み込めます。

（空欄）：通過ファイル名を表示しますが読み込めません。

／：通過ファイル名も表示せず、読み込めません。

〔注意〕

- 本機で作成したプログラムを他のPBシリーズ(PB-410、FX-720Pを除く)に移す場合は、プログラム中にREAD#、WRITE#、RESTORE#の命令があってはいけません。
  また、KEY$、MID$はPB-100シリーズではKEY、MIDとしてください。
- PB-100シリーズで作成されたプログラムを本機で実行した場合に、そのままでは正しく実行しないことがあります。
  IF～THEN分岐先で、分岐先に数式が用いられている場合→エラー
  　⇨IF～THEN GOTO分岐先に訂正

6−3 あると便利なオプション

本機は本体だけでも、かなり便利に使えますが、オプションとしてカセットテープレコーダー用インタフェイス(アダプター)<FA-3>とミニプリンタ<FP-12>または<FP-12S>があります。
<FA-3>は本機で組んだプログラムを短時間のうちに記録したり、呼び戻したりできますし、メモリー内のデータも記録しておくことができます。
<FP-12>または<FP-12S>はプログラムやデータを印字するミニプリンタで、プログラムのリスト(内容)を印字して保存したり、計算結果を印字することができます。
では、各々の機能を簡単に説明していきます。

6−3−1 プログラムやデータを保存する

プログラムやデータをカセットテープに保存するには、<FA-3>が必要です。
FA-3とテープレコーダーの接続の仕方および注意事項については、FA-3に付属の取扱説明書をご覧ください。ここでは主に、使い方について説明します。

■プログラムの記録および呼び戻し

本機にプログラムを記憶させていくと、次のプログラムを記憶させたいが、もう入らなくなることがあります。こんなときに、前のプログラムを消してしまっては、後でもう一度使いたいときに不便です。ここでFA-3が真価を発揮します。
プログラムを記録する命令は"SAVE"または"SAVE ALL"で、"SAVE"はP0だけとか、P9だけというように、1つのプログラムエリア内のプログラムだけを記録します。"SAVE ALL"はP0からP9までの10個のプログラムエリア内のプログラムを同時に記録します。

SAVE命令

MODE 0

```
READY Pn
```

→この番号のプログラムエリア内のプログラムが記録されます。

SAVE ALL命令

プログラムエリアに関係なく、P0からP9までの10個のプログラムエリア内のプログラムが記録されます。

SAVE命令とSAVE ALL命令は、単独のエリアだけのプログラムであるか、メインプログラムの他にもサブルーチンなどとしてプログラムを使っているかによって使い分けます。
SAVE、SAVE ALL命令ともマニュアルで実行します。

例） SAVE EXE
SAVE "CASIO" EXE
SAVE ALL EXE
SAVE ALL "PB" EXE

SAVEとSAVE ALLの後に続く" "でかこんだ文字は、ファイル名と呼ばれるキーワードで、記録するプログラムに個別の名前をつけておけば、後で呼び戻すときに名前を指定して呼び戻せます。ファイル名は8文字以内のアルファベットや数字などを" "でかこんで書きます。
プログラムの呼び戻しには"LOAD"または"LOAD ALL"を使います。LOAD命令とLOAD ALL命令は、記録したときにSAVEまたはSAVE ALLで記録したかにより使い分けます。

記録＼呼び戻し	LOAD	LOAD "ファイル名"	LOAD ALL	LOAD ALL "ファイル名"
SAVE	○	×	×	×
SAVE "ファイル名"	○	○	×	×
SAVE ALL	×	×	○	×
SAVE ALL "ファイル名"	×	×	○	○

※○印は呼び戻し可。
※ファイル名は同一のものとする。

例） LOAD EXE
LOAD "ファイル名" EXE
LOAD ALL EXE
LOAD ALL "ファイル名" EXE

LOAD命令とLOAD ALL命令により、プログラムを呼び戻すとき、プログラムの記録の仕方により次の表示が出ます。

記録方式	表示
SAVE	PF：
SAVE "ファイル名"	PF：ファイル名
SAVE ALL	AF：
SAVE ALL "ファイル名"	AF：ファイル名

SAVE命令により記録したプログラムをLOAD命令により呼び戻すとき、記録したプログラムエリアと呼び戻すプログラムエリアは同一でなくてもかまいません。

例）　P0のプログラムを記録

↓

P9に呼び戻す

〈ご注意〉

プログラムを記録したり、呼び戻すときにうまくいかないことがあります。そのようなときには次の点をチェックしてください。

●記録しているときに〝ERR 9〟が表示される。

〔チェックポイント〕

本体とFA-3が正しく接続されているかチェックする。

●呼び戻しているときに〝ERR 9〟が表示される。

〔チェックポイント〕

テープの保存状態が悪く、伸びているようなときは、新しいテープと交換する。

テープレコーダーのヘッドがよごれているときは、市販のクリーニングキットでヘッドをクリーニングする。

テープレコーダーのトーン調整を中位にする。

●呼び戻しているときに、エラーも何も表示されずに、戻ってこない。

〔チェックポイント〕

テープレコーダーの出力ボリウムが低いので、最高(MAX)に近い位置までボリウムを上げる。

テープレコーダーの出力規格がFA-3に合わない。

■データバンクデータの記録および呼び戻し

データバンクに記憶させたデータを他機に移したいときや、電池を交換するときには、データをテープに記録しておきたいものです。

では、データバンクデータをテープに記録してみましょう。

データバンクデータの記録には“SAVE#”を使います。

SAVE#命令は1度に全部のデータを記録できます。

SAVE#　〝ファイル名〟

8文字以内の文字

ファイル名はプログラムの記録と同じで、8文字以内の文字を〝　〟でかこみます。

例）SAVE# "MEMO" EXE

テープに記録されたデータバンクデータを呼び戻すには、"LOAD#"を使います。LOAD#命令はテープ上に記録されたデータをそのまま呼び戻し、データバンクに記憶させますので、何かが記憶されていても前のデータを消して、新たなデータを記憶します。

LOAD# "ファイル名"
8文字以内の文字

例）LOAD# "MEMO" EXE

データバンクデータを呼び戻しているとき、次のような表示が出ます。

記録方式	表　示
SAVE#	MF：
SAVE# "ファイル名"	MF：ファイル名

■データの記録、呼び戻し

プログラムにはデータがつきものですが、いちいちキーボードから入力していてはめんどうなものです。

ここでは、メモリー内のデータをテープに記録して、再び呼び戻す方法を行なってみましょう。

データを記録するには"PUT"を使います。

PUT命令はファイル名をつけて、変数名で指定します。

PUT "ファイル名"　変数1，変数2
8文字以内の文字

ファイル名はプログラムの記録と同じで、8文字以内の文字を" "でかこみます。変数名の指定は、専用文字変数($)があるときは最初に、次の2つの変数名で、"どこから"、"どこまで"の変数を記録するか指定します。

例）

専用文字変数($)とAからMまでの13個の変数の内容を記録する。

```
PUT $,A,M
```

AからZ(10)までの36個の変数の内容を、ファイル名"CASIO"で記録する。

```
PUT "CASIO" A,Z(10)
```

※メモリーが増設してある場合。

変数名の指定はどこから、どこまでのように指定しますので、〝A, Z″という指定となり、〝Z, A″のような指定はできません。
なお、変数を文字変数として使っている場合でも、〝A$, Z$″とせずに〝A, Z″とすることもできます。
データの呼び戻しには〝GET″を使います。
GET命令もファイル名をつけて、変数名で指定します。

GET 〝ファイル名″ 変数1, 変数2
（ファイル名：8文字以内の文字）

例）

専用文字変数($)とXからZまでの3個の変数にデータを呼び戻す。

```
GET $,X,Z
```

G(0)からG(99)までの変数に、ファイル名〝PB″のデータを呼び戻す。

```
GET "PB" G(0),G(99)
```

※メモリーが増設してある場合。

GET命令により、データを呼び戻しているとき、次のような表示が出ます。

記録方式	表示
PUT $, A, Z	DF:
PUT 〝ファイル名″ G, P	DF:ファイル名

6-3-2 記録を残す

プログラムを作るときに、プログラム内容を印字できれば、デバッグや内容変更にとても便利ですし、演算結果を印字して残すのも便利なものです。
ここではミニプリンタ<FP-12>または<FP-12S>を使って印字させてみましょう。
印字させるにはプリントモード(〝PRT″点灯)でキー操作を行ないます。
プリントモードは MODE 7 と押すことにより指定され、MODE 8 と押すことにより解除されます。

■プログラムの印字

プログラムの内容を印字させるには、MODE 7 と押した後、LIST命令を実行します。

例）

次のプログラムを記録させます。

```
10 INPUT A
20 PRINT A*A
30 GOTO 10
```

記憶させた後に次の操作を行ないます。

MODE 7 LIST EXE　　　印字例

```
LIST
  10 INPUT A
  20 PRINT A*A
  30 GOTO 10
```

もし、P0からP9までの10個のプログラムエリア全部の内容を印字させたいときは、

LIST ALL EXE

と操作します。
なお、印字後はMODE 8と操作してプリントモードを解除してください。

■演算結果等の印字

演算結果等を印字させるには、MODE 7と押すか、プログラム中に"MODE 7"を書き込みます。MODE 7と押しますと、その後のキー操作全てが印字されますので、必要な所だけを印字させたいときは、プログラム中に書き込んだ方が便利です。

※プログラム中に書き込む場合は、MODEキーではなく、M O D Eと押して入力します。

例)

演算結果だけを印字する。

```
10 INPUT A
20 MODE 7
30 PRINT A*A
40 MODE 8
50 GOTO 10
```

このプログラムでは、データ入力後にプリントモードを指定し、印字(表示もします)後プリントモードを解除して、再び入力に戻ります。
なお、プリントモード中ではPRINT文による表示は停止せずに、印字後次の命令に進みます。
MODE 7により印字後は、MODE 8でプリントモードを解除してください。

■データバンクデータの印字

データバンクエリアに記憶されているデータを印字させるには、MODE 7 と押した後、LIST＃命令を実行します。

例）MODE 7 LIST＃ EXE
MODE 8

印字例

```
LIST #
  1 JIHOU   →117
  2 TENKI   →177
  3 COLLECTcall→106
  4 DENPOU  →115
  5 KOKUSAI・TEL→005
    1
  6 SHINAI・TEL →104
  7 SHIGAI・TEL →105
  8 NEWS・TOKYO →03-
    540-1212
  9 NEWS・NAGOYA→052
    -320-1212
 10 NEWS・OSAKA →06-
    924-1212
```

※印字後は MODE 8 と操作して、プリントモードを解除してください。

第7章

ライブラリー

7－1 カーレース

このゲームは変化するコースを右に左にとハンドルを切り、フェンスにぶつからないように長い距離を走りぬくレースです。

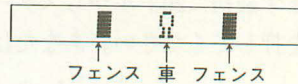
■プログラムリスト

```
10 PRINT " CAR RAC
   E !";
20 BEEP 0: GOSUB 5
   00
30 PRINT "HI-SCO:"
   ;S;"km";
40 GOSUB 500
50 X=6:Y=3:Z=9:T=0
   :C=0
60 PRINT
70 PRINT CSRY;"■";
    CSRX;"Ω"; CSRZ
   ;"■";
80 IF X= INTY THEN
    GOSUB 600
90 IF X= INTZ THEN
    GOSUB 600
100 T=T+1
110 $= KEY$
120 IF $="4" THEN X
    =X-1
130 IF $="6" THEN X
    =X+1
140 BEEP 0
150 R= RAN#*.9
160 IF RAN#>.5 THEN
     R=-R
170 IF Z+R≥12 THEN
    R=0
180 Q= RAN#*.8
190 IF RAN#>.5 THEN
     Q=-Q
200 IF Y+Q<0 THEN Q
    =0
210 IF Z-Y<3 THEN 2
    30
220 Z=Z+R:Y=Y+Q
230 GOTO 60
500 REM TIME
510 FOR U=1 TO 100:
     NEXT U
520 PRINT
530 BEEP 0
540 RETURN
600 REM CRASH
610 FOR I=1 TO 10
620 PRINT CSRX;"*";
630 BEEP 1
640 PRINT CSRX;"Ω";
650 NEXT I
660 PRINT CSR0;"<<C
    RASH !!>>";
670 GOSUB 500
680 PRINT "SCORE:";
    T*3;"km";
690 GOSUB 500
700 X=6:Y=3:Z=9
710 C=C+1
720 IF C<3 THEN RET
    URN
730 T=T*3
740 IF T>S THEN S=T
750 PRINT
760 $="GAME OVER !!
    "
770 FOR I=1 TO 12
780 PRINT MID$(I,1)
    ;: BEEP 1
790 NEXT I
800 END
```

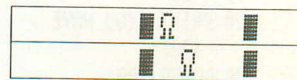
計 540steps

■ゲーム説明

使うキーは④と⑥だけで、車を左に動かすときは④キーを押し、車を右に動かすときは⑥キーを押します。

操作例	表示
RUN EXE	

左右のフェンスが動き、コースが広くなったり狭くなったりします。このフェンスにぶつからないようにうまくキーを操作してください。

左のフェンスがせまっています。
⑥

```
      ▮Ω     ▮
      ▮ Ω    ▮
```

車は3台あります。フェンスにぶつかるとクラッシュし、その時点の走行距離が表示されます。

```
<<CRASH !!>>
SCORE: 45km
```

クラッシュは2回までだいじょうぶですが、3回するとゲームオーバーとなります。

```
<<CRASH !!>>
SCORE: 234km
GAME OVER !!
```

7―2 スロットマシン

このゲームはあのラスベガスで有名なスロットマシンをゲームにしてみました。まず最初に20ドルからスタートです。かけ金を考えながらタイミングよくキーを押してください。あなたはどうかな？

■プログラムリスト

```
 10 N=20
 20 PRINT "YOU HAVE
     $20";
 30 GOSUB 500
 40 PRINT "$"; STR$
    (N);":";
 50 INPUT "IKURA",M
 60 IF N-M<0 THEN 4
    0
 70 IF M≦0 THEN 40
 80 N=N-M
 90 PRINT "$"; STR$
    (N);":"; CSR7;"
    ■ ■ ■";
100 B$="":L=0:$="♠♥
    ♥♦♦♣♣♣xx"
110 FOR I=7 TO 11 S
    TEP 2
120 PRINT CSRI;"■";
130 R= INT( RAN#*10
    +1)
140 PRINT CSRI;" ";
150 IF KEY$="" THEN
     120
160 BEEP 0
170 A$= MID$(R,1)
180 B$=B$+A$
190 PRINT CSRI;A$;
200 IF KEY$≠"" THEN
     200
210 NEXT I
220 $=B$
230 RESTORE 280
240 FOR J=3 TO 1 ST
    EP -1
250 FOR I=1 TO 4
260 READ C$: IF C$=
     MID$(1,J) THEN
     READ L: GOTO 3
    00
270 READ F: NEXT I:
     NEXT J
280 DATA ♠♠♠,30,♥♥♥
    ,25,♦♦♦,20,♣♣♣,
    15
290 DATA ♠♠,10,♥♥,8
    ,♦♦,7,♣♣,5,♠,2,
    ♥,0,♦,0,♣,0
300 IF L=0 THEN 340
310 FOR J=1 TO L*M
320 N=N+1: BEEP 1:
    PRINT CSR1; STR
    $(N);
330 NEXT J
340 GOSUB 500
350 IF N>0 THEN 40
360 PRINT "GAME OVE
    R !!";
370 END
500 REM WAIT
510 FOR Z=1 TO 200:
     NEXT Z
520 PRINT : BEEP 0
530 RETURN
```

■ゲーム説明

かけ金は1ドルから手持ちの金額まで自由にかけられます。
配当金(配当金に対する倍率)は次のようになっています。

♠♠♠	30	♦♦♦	20
♠♠−	10	♦♦−	7
♠−−	2	♣♣♣	15
♥♥♥	25	♣♣−	5
♥♥−	8		

※−印はどのマークが表示されてもかまいません。

この表示になるように、タイミングよくキーを押してください。

操作例

RUN EXE

表示

```
YOU HAVE $20
$20:IKURA?
```

現在の所持金が表示されます。

●では5ドルをかけてみましょう。

5 EXE

```
$15: ■ ■ ■
```

残り　点滅

●いずれかのキー(英文字・数字・演算キー)を押すたびに1つずつ表示が止っていきます。

\+

\+ +

```
$15: ♠ ■ ■
$15: ♠ ♣ ♠
$15:IKURA?
```

●うまくそろいませんでした。では今度は2ドルにしてみましょう。

2 EXE

\+

\+

\+

```
$13: ■ ■ ■
$13: ♣ ■ ■
$13: ♣ ♣ ■
$13: ♣ ♣ X
$23: ♣ ♣ X
```

持ち金が増えていきます。

```
$23:IKURA?
```

●うまくそろえば、どんどん増えますが、持ち金がなくなるとゲームオーバーです。

```
GAME OVER !!
```

7—3 パクパクゲーム

このゲームはキーをうまく操作してなるべく多くのエサを食べます。エサは出たり出なかったりします。
しかし、時々エサのかわりに虫が現われます。虫は逆におそってきますので、一目散に逃げだします。虫に3回つかまるとゲームオーバーとなります。

■プログラムリスト

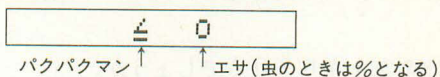
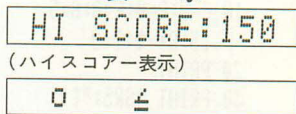
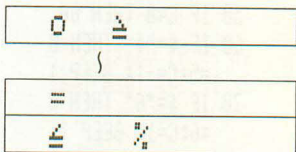
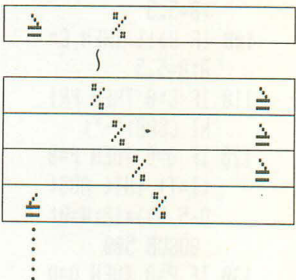
```
 10 X=5:Z$="≤":P=0:
    T=0:N=0
 20 PRINT "HI SCORE
    :"; STR$(S);
 30 FOR V=1 TO 100:
     NEXT V
 40 BEEP 1
 50 PRINT
 60 PRINT CSRX;Z$;
 70 IF P=1 THEN 140
 80 Q$="o":O=10
 90 IF RAN#>.8 THEN
     Q$="%":O=20
100 IF RAN#<.7 THEN
     200
110 BEEP 1
120 Y= INT( RAN#*12
    ): IF ABS(Y-X)<
    2 THEN 120
130 P=1
140 PRINT CSRY;Q$;:
    O=O-1
150 IF INTY=X THEN
    GOSUB 500
160 IF Q$≠"%" THEN
    190
170 IF X>Y THEN Y=Y
    +1: GOTO 190
180 Y=Y-1
190 IF O<0 THEN P=0
200 $= KEY$
210 IF $="4" THEN X
    =X-1:Z$="≥": BE
    EP 0: IF X<0 TH
    EN X=11
220 IF $="6" THEN X
    =X+1:Z$="≤": BE
    EP 0: IF X>11 T
    HEN X=0
230 GOTO 50
500 REM *CATCH*
510 BEEP 1
520 IF Q$="o" THEN
    PRINT CSRX;"=";
    :T=T+10:P=0: RE
    TURN
530 FOR I=1 TO 10
540 PRINT CSRX;"*";
    : BEEP 0
550 PRINT CSRX;"+";
    : BEEP 1
560 NEXT I
570 N=N+1: IF N<3 T
    HEN P=0: RETURN
580 PRINT
590 IF S<T THEN S=T
    : PRINT "HI ";
600 PRINT "SCORE:";
     STR$(T);
610 BEEP 1: BEEP 1:
     END
```

■ゲーム説明

●使用するキーは④キーと⑥キーだけです。

④：左に進む(左端に進むと右端から出てくる)

⑥：右に進む(右端に進むと左端から出てくる)

●表示は次のようになっています。

●エサと虫は現われた後、ある一定時間で消えます。
ではゲームをスタートさせましょう。

● 3回虫につかまるとスコアーを表示して、ゲームオーバーになります。

ハイスコアーのとき　　HI SCORE:200

ハイスコアーでないとき　　SCORE:130

★得点はエサを1個食べるごとに10点加わります。

7−4 要塞を守れ

このゲームは、左右から攻めてくる敵機から要塞を守るゲームです。
敵機は左または右のどちらかから体当りをしてきます。こちらには、1門の高射砲しかありません。この高射砲も旧式で、一発射つと次の発射まで時間がかかります。
さて、あなたは要塞を守りきれるでしょうか。

■プログラムリスト

```
 10 C=0:P=0:F=0:B=5
    .5:O=-1:T=0
 20 PRINT
 30 PRINT CSR5;"[]"
    ;
 40 $= KEY$
 50 IF C≠0 THEN 80
 60 IF $="4" THEN B
    =5:C=-1: BEEP 1
 70 IF $="6" THEN B
    =6:C=1: BEEP 1
 80 B=B+C
 90 IF B<0 THEN C=0
    :B=5.5
100 IF B>11 THEN C=
    0:B=5.5
110 IF C≠0 THEN PRI
    NT CSRB;"-";
120 IF O=B THEN P=0
    :T=T+ INT( ABS(
    O-5.5))*10:W=0:
     GOSUB 500
130 IF P≠0 THEN O=O
    + SGN(5-O): GOT
    O 170
140 IF RAN#<0.8 THE
    N 210
150 O=0:P=1
160 IF RAN#>0.5 THE
    N O=11
170 PRINT CSRO;"*";
    : BEEP 0
180 IF O=B THEN P=0
    :T=T+ INT( ABS(
    O-5.5))*10:W=0:
     GOSUB 500
190 IF O=5 THEN W=1
    : GOSUB 500
200 IF O=6 THEN W=1
    : GOSUB 500
210 GOTO 20

500 REM *OUT*
510 FOR I=1 TO 10
520 PRINT CSRO;"*";
    : BEEP W
530 PRINT CSRO;"+";
    : BEEP W
540 NEXT I
550 C=0:B=5.5:O=-1
560 IF P=0 THEN RET
    URN
570 F=F+1:P=0
580 IF F<3 THEN RET
    URN
590 PRINT : PRINT "
    SCORE:";T
600 END
```

■ゲーム説明

●使用するキーは④と⑥の２つです。

④：左に発射

⑥：右に発射

●表示は次のようになっています。

●敵機は出現と同時に、要塞にどんどん向ってきます。
では、ゲームスタート。

操 作 例　　　　**表　示**

```
        [ ]
*       [ ]
```

●敵機が左から現われました。

④

```
  *   -[ ]
    *- [ ]
    *  [ ]
```

↑命中！（＊と＋の点滅）

```
       [ ]    *
```

●今度は右から現われましたが、まちがって④キーを押してしまいました。

●一度まちがって射ったときは、弾道が消えるまで次の発射はできません。
３回体当りされるとゲームオーバーとなり得点が表示されます。

```
SCORE: 590
```

★得点はできるだけ要塞から離れた位置で敵機を射つほど高くなります。

7-5 並びかえゲーム

このゲームは不規則に並んでいる0から9の数字を正しく〝0123456789〟と並びかえタイムを競うものです。

★あなたの腕まえは？

スコアー	評　価
50未満	あなたは天才!!　運も強い。
50～69	抜群の反射神経です。
70～99	やっと人並み。もう一歩です。
100～149	まだまだ練習不足です。
150以上	何かのまちがいでしょう？　もう一度どうぞ。

■プログラムリスト

```
10 PRINT "Hi-Sc:";
   H;: FOR I=1 TO
   10: BEEP 1: NEX
   T I
20 $=""
30 PRINT : PRINT "
   << WAIT ! >>";
40 A$= STR$( INT(
   RAN#*10))
50 FOR I=1 TO LEN(
   $)
60 IF A$= MID$(I,1
   ) THEN 40
70 NEXT I
80 $=$+A$
90 IF LEN($)<10 TH
   EN 40
100 PRINT
110 X=5:B$=" ":N=0
120 PRINT CSR0;$;">
    ";B$;
130 PRINT CSRX;"*";
140 K$= KEY$
150 IF K$="4" THEN
    BEEP 0:X=X-1: I
    F X<0 THEN X=0
160 IF K$="6" THEN
    BEEP 0:X=X+1: I
    F X>9 THEN X=9
170 IF K$="+" THEN
    GOSUB 500
180 N=N+1
190 IF $≠"012345678
    9" THEN 120
200 IF H=0 THEN H=N
210 IF H>N THEN H=N
220 PRINT : PRINT "
    [";$;"]";
230 FOR I=1 TO 10
240 BEEP 0: BEEP 1
250 NEXT I
260 PRINT
270 PRINT "SCORE";N
    ;
280 END
500 BEEP 1
510 IF X=0 THEN C$=
     MID$(1,1):$=B$
    + MID$(2): GOTO
     540
520 C$= MID$(X+1,1)
530 $= MID$(1,X)+B$
    + MID$(X+2)
540 B$=C$
550 RETURN
```

使うキーは④と⑥と⊞で、④を押すとカゴが左に移動し、⑥を押すとカゴが右に移動します。⊞キーを押すと、現在のカゴの位置にある数字をカゴに入れ、カゴに入っていた数字をかわりに置きます。

表示の見方

表示されるSCORE(点数)は作業時間で、小さい程良いのです。
なお、ハイスコアーが残るようになっていますので、最初に実行するときや、他のプログラムを実行した後は、必ず"CLEAR EXE"と操作してください。

操作	表示	
CLEAR EXE		…最初だけ行ないます
RUN EXE	Hi-Sc:0	…ハイスコアー表示
	<<WAIT!>>	…問題作成中
	3721*94650>	
⊞	3721*94650>8	…8をカゴに入れる
⑥⑥⑥⑥	3721 946*0>8	…正しい位置に移動
⊞	3721 946*0>5	…8を置く。5がカゴに入る
④④④	3721 *4680>5	…5を置く。9がカゴに入る
⊞	3721 *4680>9	…正しい位置に移動
以下順に繰り返す	⋮	
⊞	0123*56789>4	…最後の4を置く
	[0123456789]	…並びかえ成功
	SCORE 132	…スコアー表示

ゲームを続ける場合は、直接"RUN EXE"と操作してください。
カゴに入れる順番やキーを押すタイミングにより、スコアーはどんどん縮まります。

7−6 陸上競技

このゲームは3つの種目からできています。

1（P0）：100m競走　　2（P1）：走り幅飛び　　3（P2）：ハードル

これ等のプログラムは独立したプログラムエリアにしてあります。

順番としてまずP0の100m競争から始め、一定レベル以上の成績を上げれば、次の競技に進めます。最後のハードルを完走した場合には総合点も表示されます。

※このゲームは全部で1533ステップを必要としますので、増設RAMパック〈OR-1Ⓔ〉をつけてから入力してください。

■プログラムリスト

```
P0
  10 X=0:Y=0:W=0:Z=0
     :K=0:B=100
  20 PRINT "HI-SCO";
     Q;"s";
  30 V=B: GOSUB #9:
     BEEP 1
  40 IF KEY$≠"" THEN
      GOSUB #7: GOTO
      30
  50 PRINT CSRX;"Ω";
      CSR11;"!";
  60 FOR I=1 TO 5
  70 IF KEY$≠"" THEN
      W=W+.2
  80 Z=Z+1
  90 NEXT I
 100 PRINT CSRX;" ";
 110 FOR I=1 TO 5
 120 IF KEY$≠"" THEN
      W=W-.2
 130 NEXT I
 140 X=X+W:W=0
 150 IF INTX≠Y THEN
     BEEP :Y= INTX
 160 IF X<0 THEN X=0
 170 IF X<11 THEN 50
 180 PRINT : BEEP 1
 190 D= RND(Z/12,-3)
 200 PRINT "TIME:";D
     ;"s";
 210 IF Q=0 THEN Q=D
 220 IF D<Q THEN Q=D
     : GOSUB #6
 230 GOSUB #9
 240 IF D≥15 THEN #8
 250 PRINT "NEXT GAM
     E ?";
 260 IF KEY$="" THEN
      260
 270 GOTO #1
```

343steps

```
P1
  10 MODE 4:K=0
  20 PRINT
  30 PRINT "HI-SCO";
     P;"m";
  40 V=B: GOSUB #9
  50 W=0
  60 BEEP 1
  70 FOR X=0 TO 11 S
     TEP .5
  80 PRINT CSRX;"Ω";
      CSR11;"_";
  90 $= KEY$
 100 IF $≤"Z" THEN I
     F $≥"A" THEN W=
     W+1
 110 IF $≥"0" THEN I
     F $≤"9" THEN GO
     TO 200
 120 V=20-W
 130 GOSUB #9: BEEP
 140 NEXT X
 150 FOR M=1 TO 5
 160 $= KEY$
 170 IF $≥"0" THEN I
     F $≤"9" THEN 20
     0
 180 NEXT M
 190 GOSUB #7: GOTO
     40
 200 BEEP 1
 210 FOR J=1 TO 80
 220 IF KEY$="" THEN
      240
 230 NEXT J
 240 BEEP
 250 R=W/2*(6-M)* CO
     S ABS(45-J)/6
 260 FOR X=0 TO R ST
     EP .5
 270 PRINT CSRX;"o";
 280 V=10: GOSUB #9
 290 BEEP 1
 300 NEXT X
 310 BEEP
 320 PRINT CSRR;"Ω";
 330 V=B: GOSUB #9
 340 E= RND(R,-3)
 350 IF E<2 THEN GOS
     UB #7: GOTO 40
```

```
360 PRINT "SCORE:";
    E;"m";
370 IF P<E THEN P=E
    : GOSUB #6
380 V=B: GOSUB #9
390 IF E<7 THEN #8
400 PRINT "NEXT GAM
    E ?";
410 IF KEY$="" THEN
     410
420 GOTO #2
```

456steps

P2

```
 10 $="    I   I
    I   I   I":X=0:
    K=0:Y=2:W=0:Z=0
 20 PRINT : PRINT "
    HI-SCO";O;"s";
 30 V=B: GOSUB #9
 40 BEEP 1
 50 IF KEY$≠"" THEN
     GOSUB #7: GOTO
     30
 60 PRINT CSR0; MID
    $(Y,11);
 70 PRINT CSRX;"Ω";
 80 Z=Z+1
 90 A$= KEY$
100 IF A$≦"9" THEN
    IF A$≧"0" THEN
    H=1: BEEP 1: PR
    INT CSR0;"μ";
110 IF Y=1 THEN Y=Y
    +1: IF H≠1 THEN
     Z=Z+3: GOSUB #
    7
120 IF A$≦"Z" THEN
    IF A$≧"A" THEN
    Y=Y+1:W=W+1: BE
    EP
130 IF Y>4 THEN Y=1
140 H=0
150 IF W<30 THEN 60
160 PRINT CSR0;"I
      I   I  I";
170 PRINT CSRX;"Ω";
180 Z=Z+1
190 A$= KEY$
200 IF A$≦"9" THEN
    IF A$≧"0" THEN
    H=1: BEEP 1: PR
    INT CSRX;"μ";
210 IF FRAC(X/4)=0
    THEN X=X+1: IF
    H≠1 THEN Z=Z+3:
     GOSUB #7
220 IF A$≦"Z" THEN
    IF A$≧"A" THEN
    X=X+1: BEEP
230 H=0
240 IF X<12 THEN 16
    0
250 BEEP : PRINT
260 F= RND(Z/1.1,-2
    )
270 PRINT "TIME:";F
    ;"s";
280 IF O=0 THEN O=F
290 IF F<O THEN O=F
    : GOSUB #6
300 GOSUB #9
310 IF F>60 THEN #8
320 R= INT( COS(D*3
    )*200+ SIN(E*3)
    *200+ COS(F*3)*
    200)
330 PRINT "TOTAL SC
    ORE";R;" points
    ";
340 IF T=0 THEN T=R
350 IF T<R THEN T=R
    : GOSUB #6
```

603steps

P6

```
 10 FOR I=1 TO 10:
    BEEP 1: BEEP :
    NEXT I
 20 RETURN
```

22steps

P7

```
 10 PRINT : PRINT "
    FOUL !!";: BEEP
     : BEEP
 20 IF K=0 THEN K=1
    : RETURN
 30 GOTO #8
```

39steps

P8

```
 10 PRINT
 20 $="GAME OVER !!
    "
 30 FOR I=1 TO 12
 40 PRINT MID$(I,1)
    ;: BEEP
 50 NEXT I
```

20steps

P9

```
 10 FOR U=1 TO V: N
    EXT U
 20 PRINT
 30 RETURN
```

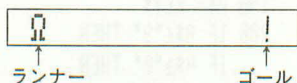
50steps

計 1533steps

P0：100m競争	P6：BEEP効果音	P9：一定時間停止用
P1：走り幅飛び	P7：ファウル処理	
P2：ハードル	P8：ゲームオーバー処理	

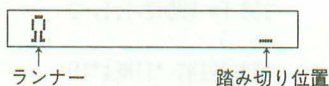
■ゲーム説明

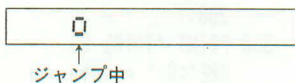
RUN EXE または SHIFT P0 でスタートさせます。

100m競争

ランナーが点滅しますので、表示しているときにいずれかのキーを押すと前進(右に進む)します。ランナーが消えているときに押すと後退(左に進む)します。

"TIME"(経過時間)が15秒以内ですと、次の走り幅飛びに進めます。

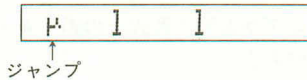
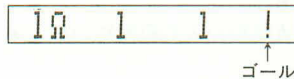
なお、ランナーが表示される前にキーを押しますと「FOUL!!」(ファウル)となり、やり直しとなります。ファウルは1回まで許されますが、2回行なうとゲームオーバーとなります。

走り幅飛び

アルファベットキー(A~Z)を押すとランナーの速度が増します。キーを押さなくても進みますが、加速がつかないとジャンプに失敗します。

ランナーが踏み切り位置にきたときに、数値キー(0~9)を押します。このとき、数字キーを押している時間により飛び出す角度がかわりますので、押す時間は短かすぎても長すぎてもいけません。

ジャンプした後はランナーが飛びます。踏み切り位置をすぎてもジャンプしないときや、ジャンプに失敗したときはファウル(FOUL!!表示)となり、1回まで許されます。

なお、ジャンプした距離が7m以下の場合は失格となり、次のハードルには進めません。

ハードル

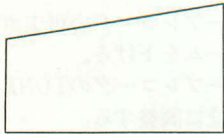
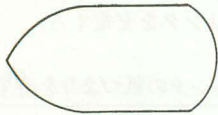
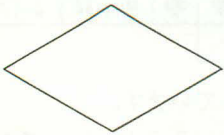
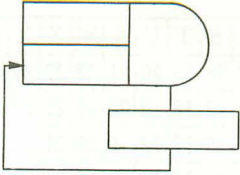
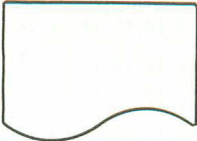
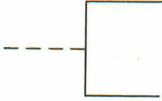
まず，アルファベットキーを押し続けてランナーを走らせます。ランナーがハードルに近づいたらタイミング良く数字キーを押してジャンプさせます。

ハードルをいくつか飛びこえていくと、ゴールが見えてきます。

ランナーが表示される前にスタートしたり、ハードルをたおしたりするとファウルとなります。ファウルは1回まで許されます。
経過時間が60秒以上ですと、失格となり総合得点は表示されません。

●使うキー

100m競争で使うキーは[MODE]、[⇦]、[⇨]、[SHIFT]、[FUNC]、[AC]、[DEL]、[STOP]、[EXE]、[MEMO]以外なら、どのキーを押してもかまいません。
走り幅飛びとハードルでは、ランナーが走るときは[A]～[Z]のアルファベットキーならどれでもかまいません。また、ジャンプするときは[0]～[9]の数字キーならどれでもかまいません。

エラーメッセージ一覧表

エラーコード	意味	エラー原因	対策
1	メモリーオーバーまたはシステムスタックオーバー	●ステップ数不足でプログラムが書き込めない。またはメモリーが増設できない。 ●計算式が複雑すぎてスタックオーバーしている。	●不要なプログラムをクリアするか、メモリー数を減らす。 ●数式を分割して簡単にする。
2	構文エラー	●プログラム等に書式上の誤りがある。 ●代入文等で左辺の型式と右辺の型式が異なる。	●入力したプログラム等の誤りを修正する。
3	数学的エラー	●数式の演算結果が10^{100}以上の場合。 ●数値関数の入力範囲外の場合。 ●結果が不定または不能となる場合。	●演算式またはデータを修正する。 ●データを判断する。
4	未定義エラー	●GOTO文、GOSUB文の指定行番号がない。 ●READ文またはREAD＃文に対するデータが不足している。	●指定行番号を修正する。 ●データ数を正しくする。
5	引数エラー	●引数を必要とするコマンド、関数において、引数が入力範囲外の場合。	●引数の誤りを修正する。
6	変数エラー	●増設されていないメモリーを使おうとした。 ●同一メモリーを数値変数と文字変数に同時に使おうとした。	●適切にメモリーを増設する。 ●同時に同一メモリーを文字変数、数値変数として使わないようにする。
7	ネスティングエラー	●サブルーチン実行中以外でRETURN文が出てきた場合。 ●FORループ中以外でNEXT文が出てきたり、FOR文に対するNEXT文の変数が異なる場合。 ●サブルーチンのネスティングが8段をこえた場合。 ●FORループのネスティングが4段をこえた場合。	●不必要なRETURN文やNEXT文を取る。 ●サブルーチンやFOR・NEXTループをレベル内にする。

エラーコード	意味	エラー原因	対策
8	パスワードエラー	●パスワードが設定されているときに、異なるパスワードを入力した。 ●パスワードが設定されているのに、LISTやNEWなどの禁止コマンドを実行した。	●正しいパスワードを入力して、パスワードを解除する。
9	オプションエラー	●テープレコーダが接続されていないのに、SAVEまたはPUTコマンドを実行した場合。 ●LOADまたはGETコマンドにより入力された信号がわれており、読み込めない場合。 ●プリンタの充電が十分でない場合。 ●プリンタの紙づまり。	●テープレコーダを接続する。 ●テープレコーダの再生ボリュームを下げる。 ●テープレコーダのTONEを中位に調整する。 ●テープをかえる。 ●テープレコーダのヘッドをクリーニングする。 ●プリンタを充電する。 ●プリンタの紙づまりを直す。

キャラクター表

	スペース	+	−	＊	／	↑	!	”	#	$	>	≧	=	≦	<	≠
数字	0	1	2	3	4	5	6	7	8	9	.	π	)	(	Ē	E
英大文字	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	Q	R	S	T	U	V	W	X	Y	Z						
英小文字	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
	q	s	r	t	u	v	w	x	y	z						
記号	?	,	;	:												
グラフィック記号	○	Σ	°	△	@	×	÷	♠	←	♥	♦	♣	μ	Ω	↓	→
	%	¥	□	[	&	_	’	•	]	■	＼					

表は左から右へ、次の段の左から右へと小さい順に並んでいます。
一番小さいキャラクターは、スペースで、一番大きいキャラクターは、グラフィク記号の＼（SHIFT P と押すと表示される）です。

フローチャートの主な記号

記号	名称	意味
	端子	開始、終了等
	手操作入力	キーボードからのデータ入力
	出力	出力の機能
	処理	あらゆる種類の処理機能
	定義済み処理	サブルーチンなど、別の場所で定義されている命令群
	判断	いくつかの択一的径路のうち、どの径路をとらせるかを決める判断

記　　号	名　　称	意　　味
	FOR・NEXTループ	FOR文とNEXT文の間で、一定回数分処理を行なう。
	書　　類	書類を媒体とする入出力機能
	流　れ　線	記号を結びつける機能
	結　合　子	フローチャートのほかの場所への出口またはほかの入口
	注　　釈	明瞭にするため、説明または注意を加える機能

配列変数表

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
B	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
C	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
D	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
E	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
F	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
G	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
H	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
I	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
J	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
K	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
M	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12	13
N	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11	12
O	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10	11
P	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9	10
Q	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8	9
R	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7	8
S	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6	7
T	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5	6
U	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4	5
V	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3	4
W	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2	3
X	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1	2
Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	1
Z	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

表の見方

縦に並んでいるA～Zを配列変数名として使用した場合、通常のA～Zの変数とどこから重なるかを見ます。

例）H(0)～H(9)→H～Q

規　格

型　　　式：PB-110

基本計算機能：負数、指数、カッコを含む四則計算(加減・乗除の優先順位判別機能つき)

組込関数機能：三角・逆三角関数(角度単位は度・ラジアン・グラジアン)、対数・指数関数、開平、べき乗、整数化、整数部除去、絶対値、符号化、有効桁数指定、小数点以下指定、10↔60進変換、乱数、π

コ マ ン ド：INPUT、PRINT、GOTO、ON～GOTO、FOR・NEXT、IF～THEN、GOSUB、ON～GOSUB、RETURN、READ、DATA、RESTORE、STOP、END、RUN、LIST、LIST ALL、MODE、SET、CLEAR、NEW、NEW ALL、DEFM、PASS、REM、BEEP、LET、SAVE、SAVE ALL、LOAD、LOAD ALL、PUT、GET、VERIFY
NEW#、LIST#、LOAD#、SAVE#、READ#、WRITE#、RESTORE#

プログラム関数：KEY$、CSR、LEN、MID$、VAL、STR$

計 算 範 囲：$\pm 1\times 10^{-99}\sim\pm 9.999999999\times 10^{99}$および0、内部演算は仮数部12桁を使用

プログラム言語：BASIC(ベーシック)

ステップ数：544ステップ(OR-1Ⓔ増設時1568ステップ)

組込プログラム数：最大10組(P0～P9)

メモリー数：標準26メモリー、および専用文字変数($)

スタック数：サブルーチン　8段・FOR・NEXTループ　4段
数値　6段・演 算 子　12段

表示桁数および内容：仮数部10桁(負符号含む)、または仮数部8桁(負数7桁)＋指数部2桁、内容　EXT、Ⓢ、RUN、WRT、DEG、RAD、GRA、TR、PRT、STOPの各状態表示付

表 示 素 子：12桁ドットマトリクス液晶

主 要 素 子：C-MOS VLSI他

電　　　源：リチウム電池(CR-2032) 2個使用

消 費 電 力：最大0.03W

電 池 寿 命（連続使用）：本体のみ 約140時間
オプション使用時 約70時間

オートパワーオフ：約6分

使 用 温 度：0℃～40℃

大きさ・重さ：幅165 奥行71 高さ9.8mm、119g(電池込み)

付 属 品：ソフトケース

コマンド索引

カシオ計算機株式会社営業本部

東京都新宿区西新宿2－6　新宿住友ビル
(〒160)　☎03-347-4811(代表)

カシオ計算機サービスセンター

地区	電話	郵便番号	住所
旭川	0166-23-8580	〒070	旭川市七条通り8丁目
札幌	011-231-2343	〒060	札幌市中央区南一条西12丁目
釧路	0154-24-8575	〒085	釧路市光陽町6－1
青森	0177-22-7466	〒030	青森市勝田2－1－12
秋田	0188-33-6211	〒010	秋田市中通り6－1－55
盛岡	0196-24-2502	〒020	盛岡市本町通り3－19－6
仙台	0222-27-1404	〒980	仙台市一番町2－3－32
山形	0236-42-8018	〒990	山形市あこや町3－14－39
郡山	0249-33-5172	〒963	福島県郡山市香久池2－11－6
宇都宮	0286-34-0395	〒320	宇都宮市西大寛2－1－3
前橋	0272-53-3000	〒371	前橋市元総社町92－5
水戸	0292-25-6985	〒310	水戸市中央1－2－20
埼玉	0486-66-8567	〒330	大宮市大成町4－83
千葉	0472-43-1751	〒260	千葉市登戸町2－276
東京	03－862-4141	〒101	千代田区神田佐久間町2－23
中央	03－583-4111	〒106	港区六本木2－3－6
城南	03－787-3721	〒145	大田区上池台1－1－6
城西	03－376-3221	〒160	新宿区西新宿4－2－18
多摩	0425-23-3531	〒190	立川市錦町3－2－25
横浜	045-211-0821	〒231	横浜市中区弁天通り6－85
新潟	0252-41-4105	〒950	新潟市米山3－1－5
長野	0262-28-9360	〒380	長野市岡田町30－20
甲府	0552-37-6371	〒400	甲府市城東2－22－11
静岡	0542-81-8085	〒420	静岡市西中原1－4－35
浜松	0534-64-1658	〒435	浜松市西塚町324
豊橋	0532-53-2515	〒440	豊橋市魚町5
名古屋	052-263-0454	〒460	名古屋市中区栄4－6－15
岐阜	0582-62-0145	〒500	岐阜市鷹見町8
三重	0592-27-5066	〒514	津市鳥居町191
富山	0764-22-2251	〒930	富山市白銀町2－1
金沢	0762-37-8511	〒920	金沢市諸江町下丁93－1
京都	075-351-1161	〒600	京都市下京区五条通り堀川東入ル
大阪	06－362-8181	〒530	大阪市北区南森町2－1－20
和歌山	0734-31-7807	〒640	和歌山市九家の丁5
神戸	078-392-4123	〒650	神戸市中央区北長狭通り4－4－18
岡山	0862-41-8471	〒700	岡山市西古松西町8－21
福山	0849-24-2830	〒720	福山市南本庄町2－130
広島	082-263-1090	〒730	広島市南区稲荷町4－1
山口	0835-22-6164	〒747	防府市戎町1－10－16
高松	0878-62-5240	〒760	高松市亀岡町9－16
松山	0899-45-2234	〒790	松山市平和通り1－1－5
福岡	092-411-2684	〒812	福岡市博多区博多駅南1－2－15
長崎	0958-61-8084	〒852	長崎市宝栄町2－26
熊本	096-367-0650	〒862	熊本市健軍4－1－5
鹿児島	0992-56-3575	〒890	鹿児島市上荒田町30－18